

**T. C.  
HACETTEPE ÜNİVERSİTESİ  
SAĞLIK BİLİMLERİ ENSTİTÜSÜ**

**MEDYAL PREFRONTAL KORTEKSTE  
SİNAPTİK BAĞLANTI REORGANİZASYONU  
STRESE DUYARLILIĞIN BİR BELİRLEYİCİSİ OLABİLİR Mİ?**

**Dr. Sabahaddin Taha SOLAKOĞLU**

**Temel Nörolojik Bilimler Programı  
DOKTORA TEZİ**

**ANKARA  
2024**



**T. C.  
HACETTEPE ÜNİVERSİTESİ  
SAĞLIK BİLİMLERİ ENSTİTÜSÜ**

**MEDYAL PREFRONTAL KORTEKSTE  
SİNAPTİK BAĞLANTI REORGANİZASYONU  
STRESE DUYARLILIĞIN BİR BELİRLEYİCİSİ OLABİLİR Mİ?**

**Dr. Sabahaddin Taha SOLAKOĞLU**

**Temel Nörolojik Bilimler Programı  
DOKTORA TEZİ**

**TEZ DANIŞMANI  
Prof. Dr. Emine EREN-KOÇAK**

**İKİNCİ DANIŞMAN  
Doç. Dr. Şefik Evren ERDENER**

**ANKARA**

**2024**

**Medyal Prefrontal Kortekste Sinaptik Bağlantı Reorganizasyonu  
Strese Duyarlılığın Bir Belirleyicisi Olabilir mi?  
Dr. Sabahaddin Taha Solakođlu**

**Danışman: Prof. Dr. Emine Eren-Koçak  
İkinci Danışman: Doç. Dr. Şefik Evren Erdener**

Bu tez çalışması 06.12.2024 tarihinde jürimiz tarafından “Temel Nörolojik Bilimler Programı” nda doktora tezi olarak kabul edilmiştir.

<b>Jüri Başkanı:</b>	Prof. Dr. Aygün Ertuđrul (Hacettepe Üniversitesi)
<b>Üye:</b>	Prof. Dr. Yavuz Ayhan (Hacettepe Üniversitesi)
<b>Üye:</b>	Doç. Dr. Gül Yalçın-Çakmaklı (Hacettepe Üniversitesi)
<b>Üye:</b>	Prof. Dr. İpek Yalçın-Christmann (CNRS, Fransa)
<b>Üye:</b>	Doç. Dr. Hale Yapıcı-Eser (Koç Üniversitesi)

Bu tez, Hacettepe Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca yukarıdaki jüri tarafından uygun bulunmuştur.

19 Aralık 2024

Prof. Dr. Müge Yemişçi Özkan  
Enstitü Müdürü

## YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe Üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alınarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan **“Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge”** kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H.Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- Enstitü / Fakülte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir. <sup>(1)</sup>
- Enstitü / Fakülte yönetim kurulunun gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihinden itibaren ... ay ertelenmiştir. <sup>(2)</sup>
- Tezimle ilgili gizlilik kararı verilmiştir. <sup>(3)</sup>

26 / 12 / 2024

Sabahaddin Taha Solakoğlu

<sup>1</sup>“Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge”

- (1) Madde 6. 1. Lisansüstü tezle ilgili patent başvurusu yapılması veya patent alma sürecinin devam etmesi durumunda, tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulu iki yıl süre ile tezin erişime açılmasının ertelenmesine karar verebilir.
- (2) Madde 6. 2. Yeni teknik, materyal ve metotların kullanıldığı, henüz makaleye dönüşmemiş veya patent gibi yöntemlerle korunmamış ve internette paylaşılması durumunda 3. şahıslara veya kurumlara haksız kazanç imkanı oluşturabilecek bilgi ve bulguları içeren tezler hakkında tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulunun gerekçeli kararı ile altı ayı aşmamak üzere tezin erişime açılması engellenebilir.
- (3) Madde 7. 1. Ulusal çıkarları veya güvenliği ilgilendiren, emniyet, istihbarat, savunma ve güvenlik, sağlık vb. konulara ilişkin lisansüstü tezlerle ilgili gizlilik kararı, tezin yapıldığı kurum tarafından verilir \*. Kurum ve kuruluşlarla yapılan işbirliği protokolü çerçevesinde hazırlanan lisansüstü tezlere ilişkin gizlilik kararı ise, ilgili kurum ve kuruluşun önerisi ile enstitü veya fakültenin uygun görüşü üzerine üniversite yönetim kurulu tarafından verilir. Gizlilik kararı verilen tezler Yükseköğretim Kuruluna bildirilir.  
Madde 7.2. Gizlilik kararı verilen tezler gizlilik süresince enstitü veya fakülte tarafından gizlilik kuralları çerçevesinde muhafaza edilir, gizlilik kararının kaldırılması halinde Tez Otomasyon Sistemine yüklenir.

## ETİK BEYAN

Bu çalışmadaki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi, görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu, kullandığım verilerde herhangi bir tahrifat yapmadığımı, yararlandığım kaynaklara bilimsel normlara uygun olarak atıfta bulunduğumu, tezimin kaynak gösterilen durumlar dışında özgün olduğunu, Prof. Dr. Emine Eren-Koçak ve Doç. Dr. Şefik Evren Erdener danışmanlığında tarafımdan üretildiğini ve Hacettepe Üniversitesi Sağlık Bilimleri Enstitüsü Tez Yazım Yönergesine göre yazıldığını beyan ederim.

Dr. Sabahaddin Taha Solakoğlu

## TEŞEKKÜR

İlk günden bugüne, herhangi bir koşulda her türlü desteği bana karşılıksız şekilde sağlamış ve desteklerinin devam etmesini umduğum annem Şükriye Solakoğlu, babam Seyit Kamil Solakoğlu ve kardeşim Sinem'e çok teşekkür ederim. Bu tezi onlara adıyorum.

Çalışmaya başladığımız ilk günden bugüne sinirbilimi sevdiren, iyi ve rekabetçi bir bilim insanı olmam için sürekli destek olan, bu riskli çalışmayı yapmam için beni sürekli cesaretlendiren tez danışmanlarım Prof. Dr. Emine Eren-Koçak ve Doç. Dr. Şefik Evren Erdener'e teşekkür ederim.

Tıp-Bilim Doktorluğu programının ilk gününden itibaren sinirbilimi bana sevdiren, sonrasında da bu tezi yapmam için beni teşvik eden ve tezin ilerleyişini önerileriyle sürekli takip eden Prof. Dr. Turgay Dalkara'ya teşekkür ederim.

Doktora tez başlangıcından itibaren değerli önerileriyle tezimin iyileşmesini sağlayan Doç. Dr. Hale Yapıcı-Eser'e ve önerileriyle daha farklı çalışmaların önünü açan Prof. Dr. İpek Yalçın-Christmann, Prof. Dr. Yavuz Ayhan, Doç. Dr. Gül Yalçın-Çakmaklı ve Doç. Dr. Şahin Hanalioğlu'na teşekkür ederim.

Görüntü işleme ve veri analizlerinde yardımcı oldukları için Dr. Uygur Sümbül ve Dr. Olga Gliko'ya teşekkür ederim. Onların halen devam eden destekleri tezimin bu düzeye gelmesinde etkili oldu. Hem görüntü alım sürecindeki hem de makaledeki şekillerin oluşturulmasındaki yardımları için Prof. Dr. Alp Can'a teşekkür ederim.

Enstitüdeki çalışmalarına her yolla eşlik eden, son dakikaya kadar yılmadan bu çalışmamı bitirmemi sağlayan, ancak kısıtlamalar sebebiyle haklarında tek tek not yazamadığım, Prof. Dr. Bülent Elibol, Prof. Dr. Müge Yemişçi-Özkan, Prof. Dr. Hülya Karataş-Kurşun, Dr. Aslıhan Bahadır-Varol, Dr. Nevin Belder, Dr. Buket Demir, Dr. Canan Çakır-Aktaş, Dr. Melike Sever-Bahçekapılı, Dr. Doğa Başaran, Dr. Kaan Kutay Özmen, Dr. Tuğçe Bilbay, Bilge Köse, Dr. Bengisu Solgun, Dr. Gökçe Gürler, Mesut Fırat, Doç. Dr. Dilek Ünal, Prof. Dr. Eda Derle-Çiftçi, Doç. Dr. Duygu Uygun, Dr. Onur Çağın Gürlek, Dr. Emre Cem Esen, Dr. Gülce Küreli, Dr. Ecehan Erk, Dr. Zeynep Kaya-Doğu, Dr. Kadir Soylu, Dilan Bozanoğlu, Beyza Türken, Serenay Karataş, Meltem Anlı, Nursel İlikli, Hülya Erener, Tuğba Sarıtaş, Zafer Akdoğan, İbrahim Bulut, Ali Osman Çetin'e teşekkür ederim. Çalışmaya sağladıkları proje desteği için Bilimsel Araştırma Projeleri Koordinasyon Birimine teşekkür ederim.

## ÖZET

**Solakođlu, S.T., Medyal Prefrontal Kortekste Sinaptik Bađlantı Reorganizasyonu Strese Duyarlılıđın Bir Belirleyicisi Olabilir mi?, Hacettepe Üniversitesi Sađlık Bilimleri Enstitüsü Temel Nörolojik Bilimler Programı Doktora Tezi, Ankara, 2024.** Stres yanıtı, sürekli deđişen bir çevreye uyum sađlamak için önemlidir. Ancak, bazı bireyleri stres etkisine duyarlı kılan mekanizmalar henüz tam olarak anlaşılamamıştır. Kronik stresin medial prefrontal korteks (mPFK) üzerinde dendritik atrofi ve çıkıntı kaybına neden olduđu bilinmektedir, ancak mPFK'ye ulaşan uzun menzilli projeksiyonlar tarafından oluşturulan sinapsların etkisi hala bilinmemektedir. Bu çalışmada, fare mPFK dendritlerinde ventral hipokampus (VH), bazolateral amigdala (BLA) ve ventral tegmental alan (VTA) uzun menzilli afferentler tarafından oluşturulan sinapslar, farklı renkli eGRASP yapıları kullanılarak etiketlendi. Tüm kortikal katmanları kapsayan üç boyutlu multispektral görüntüler elde edildi ve otomatize şekilde dendritleri ve sinapsları segmentlere ayrıldı. Stres uygulanmamış, strese dirençli ve stres duyarlı farelerde VH-mPFK, BLA-mPFK ve VTA-mPFK sinapslarının göreceli bolluklarını ve uzaysal organizasyonlarını kortikal katman bazında incelendi. Çalışma bulguları, farklı stres duyarlılık fenotipleri arasında sinapsların afferent ve kortikal katmana özgü farklı bir şekilde yeniden düzenlendiğini göstermekte ve bu da farklı uzun menzilli projeksiyonların stres tepkisini düzenlemede özel rolleri olduğunu öne sürmektedir.

**Anahtar Kelimeler:** Medyal Prefrontal Korteks, Mikrodevre, Depresyon, eGRASP, Strese Yatkınlık, Bölgesel Dendritik İşleme, Sinaps

Bu çalışma Hacettepe Üniversitesi (TSA-2020-18753) tarafından desteklenmiştir.



## ABSTRACT

**Solakoglu, S. T., Could Synaptic Connection Reorganization in the Medial Prefrontal Cortex Be A Determinant of Stress Susceptibility?, Hacettepe University Graduate School of Health Sciences, Doctor of Philosophy Thesis in Basic Neurologic Sciences (Neuroscience), Ankara, 2024.** Stress response is essential for adapting to an ever-changing environment. However, the mechanisms that render some individuals susceptible to stress are poorly understood. While chronic stress is known to induce dendritic atrophy and spine loss in medial prefrontal cortex (mPFC), its impact on synapses made by long-range projections terminating on the mPFC remains unknown. Here, we labeled synapses on mouse mPFC dendrites formed by ventral hippocampus (VH), basolateral amygdala (BLA) and ventral tegmental area (VTA) long-range afferents using different-colored eGRASP constructs. We obtained multispectral 3D images of the mPFC covering all cortical laminae, and automatically segmented the dendrites and synapses. We investigated the relative abundances and spatial organizations of VH-mPFC, BLA-mPFC and VTA-mPFC synapses on a laminar basis in stress naïve, stress resilient and stress susceptible mice. Our findings demonstrate afferent- and lamina-specific differential reorganization of synapses between different stress phenotypes, suggesting specific roles for different long-range projections in mediating the stress response.

**Key Words:** Medial Prefrontal Cortex, Microcircuitry, Depression, eGRASP, Stress Susceptibility, Local Dendritic Computation, Synapse

This study is supported by Hacettepe University (TSA-2020-18753).

**İÇİNDEKİLER**

ONAY SAYFASI	iii
YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI	iv
ETİK BEYAN	v
TEŞEKKÜR	vi
ÖZET	vii
ABSTRACT	viii
İÇİNDEKİLER	ix
SİMGELER VE KISALTMALAR	xii
ŞEKİLLER	xiv
TABLolar	xvii
<b>1. GİRİŞ</b>	<b>17</b>
<b>2. GENEL BİLGİLER</b>	<b>5</b>
2.1. Medyal Prefrontal Korteks Anatomisi ve Devreleri	5
2.2. Depresyon ve Nöral Devreler	7
2.3. Dendritler, Sinaptik Plastisite ve Dendritik Bilgi İşleme	11
2.4. Medyal Prefrontal Kortekste Dendritik İşleme	13
2.5. Depresyon ve Dendritler	15
<b>3. GEREÇ VE YÖNTEM</b>	<b>17</b>
3.1. Fareler	17
3.2. Virüsler	17
3.3. İntrakraniyel Enjeksiyonlar	19
3.4. Deney Takvimi	22
3.5. Kronik Sosyal Yenilgi Stresi (KSYS)	23
3.5.1. KSYS İçin Saldırgan Fare Seçimi	23
3.5.2. KSYS Stres Uygulaması	23

3.6. Sosyal Etkileşim Testi (SET)	26
3.6.1. SET Test Düzenegi	26
3.6.2. SET'in Deęerlendirilmesi	27
3.7. Sakrifikasyon ve Doku Hazırlığı	28
3.8. Görüntüleme	28
3.8.1. Sinaptik Sinyallerin Doğrulanması	28
3.8.2. Sinaptik Sinyallerin k-Ortalamalar Yöntemiyle Ayrıştırılması için Çerçeve Sıralı Görüntü Alımı	31
3.8.3. Kesit Yüzeyinden Derine Doğru Çerçeve Sıralı Görüntü Alımı	34
3.8.4. Kiremit Taramalı Görüntüleme	35
3.8.5. Reflektans Görüntüleme	37
3.9. Görüntü İşleme	37
3.9.1. Görüntü Birleştirme	39
3.9.2. Spektral Ayrıştırma	39
3.9.3. Dekonvolüsyon	40
3.9.4. Dendritik Sinyallerin Segmentasyonu ve Rekonstrüksiyonu	41
3.9.5. Dendritik Segment Tanımı	41
3.10. Sinaptik Sinyallerin Yarı Otomatik Şekilde Tespit Edilmesi	42
3.10.1. Synapse Quantifier Algoritması	42
3.10.2. Sinaptik Sinyallerin Synapse Quantifier Algoritması İçin Hazırlanması	43
3.10.3. Synapse Quantifier Uygulaması	45
3.10.4. Synapse Quantifier Algoritmasının Elle Sayımla Karşılaştırılması	46
3.11. Kortikal Katmanların Tanımlanması ve Rekonstrüksiyonlarının Hazırlanması	49
3.12. Her Bölge için Sinaps Yüzde Hesabı	49
3.13. Boyut İndirgeme ( <i>Dimensionality Reduction</i> )	50
3.14. BLA ve VH Sinapsları İçin En Yakın Uzaysal Mesafe Matrislerin Oluşturulması ve İncelenmesi	50
3.15. İstatistik Analiz	51
<b>4. BULGULAR</b>	<b>52</b>

4.1. mPFK piramidal nöron dendritlerinde BLA, VH ve VTA piramidal nöronlarından kaynaklı girdiler	52
4.2. BLA-mPFK sinaps oranlarında kortikal katman ve stres fenotipi bağımlı değişiklikler	55
4.3. VH-mPFK sinaps oranlarında kortikal katman ve stres fenotipi bağımlı değişiklikler	58
4.4. VTA-mPFK sinaps oranlarında kortikal katman ve stres fenotipi bağımlı değişiklikler	60
4.5. BLA-mPFK, VH-mPFK ve VTA-mPFK sinapslarının kortikal kolon boyunca görelî miktarlarının düşük boyuttaki temsilleriyle SDuy farelerin SDir farelerden ayırt edilmesi	62
4.6. Birden fazla bölgeden sinaptik girdi alan dendritlerin miktarının SDuy farelerde arttığı gözlemlendi	63
4.7. mPFK derin katmanlarındaki BLA-mPFK ve VH-mPFK sinapslarının uzaysal organizasyonu SDuy farelerde bozulmuştur	68
<b>5. TARTIŞMA</b>	<b>76</b>
<b>6. SONUÇLAR VE ÖNERİLER</b>	<b>83</b>
<b>7. KAYNAKLAR</b>	<b>85</b>
<b>8. EKLER</b>	<b>95</b>
EK-1: Görüntü İşleme Sırasında Kullanılan Algoritmalar	
EK-2: Veri Analizi Sırasında Kullanılan Algoritmalar	
EK-3: Tez Çalışması ile İlgili Etik Kurul İzinleri	
EK-4: Tez Çalışması Orijinallik Raporu	
<b>9. ÖZGEÇMİŞ</b>	<b>182</b>

## SİMGELER VE KISALTMALAR

<b><math>\alpha</math>2-AR</b>	$\alpha$ 2 adrenerjik reseptör
<b>AAV</b>	Adeno asosiye virüs
<b>AId</b>	Dorsal agranular insular korteks
<b>AIv</b>	Ventral agranular insular korteks
<b>akt</b>	Aktivasyon
<b>ASK</b>	Anterior singulat korteks
<b>BLA</b>	Bazolateral amigdala
<b>BOLD</b>	<i>Blood-oxygen-level-dependent</i>
<b>CaMKII<math>\alpha</math></b>	Kalsiyum kalmodulin kinaz II $\alpha$
<b>CRACM</b>	<i>Channelrhodopsin assisted circuit mapping</i>
<b>CT</b>	Kortikotalamik nöronlar
<b>ÇA</b>	Çeyrekler arası açıklık
<b>dmPFC</b>	Dorsal medyal prefrontal korteks
<b>dop</b>	Dopaminerjik nöron
<b>eGRASP</b>	<i>enhanced green fluorescent protein reconstitution across synaptic partners</i>
<b>fMRG</b>	Fonksiyonel manyetik rezonans görüntüleme
<b>GA</b>	Görüntüleme alanı
<b>GPU</b>	Grafik işlemci
<b>HCN</b>	Hiperpolarizasyonla aktive katyon kanalları
<b>HPA</b>	Hipotalamopitüiter eksen
<b>IL</b>	İnfralimbik korteks
<b>inh</b>	İnhibisyon
<b>IT</b>	İntertelensefalik nöron
<b>Kemo</b>	Kemogenetik
<b>kPFC</b>	Kontralateral prefrontal korteks
<b>KSYS</b>	Kronik sosyal yenilgi stresi
<b>LTD</b>	Uzun süreli depresyon
<b>LTP</b>	Uzun süreli güçlendirme
<b>LUMoS</b>	<i>Learning Unsupervised Means of Spectra</i>
<b>MD</b>	Medyodorsal talamus
<b>MWU</b>	Mann-Whitney U testi

<b>mGluR</b>	Metabotropik glutamat reseptörleri
<b>mPFK</b>	Medyal prefrontal korteks
<b>mORB</b>	Medyal orbitofrontal korteks
<b>NA</b>	Nümerik apertür
<b>NAk</b>	Akümbens çekirdek
<b>Opto</b>	Optogenetik
<b>ORBl</b>	Lateral orbitofrontal korteks
<b>ORBvl</b>	Ventrolateral orbitofrontal korteks
<b>PBS</b>	Fosfat tamponlu salin
<b>pir</b>	Piramidal nöron
<b>PFA</b>	Paraformaldehit
<b>PFK</b>	Prefrontal korteks
<b>PL</b>	Prelimbik korteks
<b>PSF</b>	Nokta dağılım fonksiyonu
<b>PT</b>	Piramidal yol nöronları
<b>RAM</b>	Merkezi hafıza
<b>RL</b>	Richardson-Lucy dekonvolüsyonu
<b>ROS</b>	Rastgele orman sınıflaması
<b>SDir</b>	Stres dirençli
<b>SDuy</b>	Stres duyarlı
<b>SET</b>	Sosyal Etkileşim Testi
<b>sMK</b>	Sekonder motor korteks
<b>SN</b>	Stres naif
<b>TB</b>	Temel bileşen
<b>TBA</b>	Temel bileşen analizi
<b>UMAP</b>	<i>Uniform Manifold Approximation and Projection</i>
<b>VH</b>	Ventral hipokampüs
<b>vIPFK</b>	Ventrolateral prefrontal korteks
<b>VM</b>	Ventromedyal talamus
<b>vmPFK</b>	Ventral medyal prefrontal korteks
<b>VTA</b>	Ventral tegmental alan

## ŞEKİLLER

Şekil	Sayfa
2.1. Fare mPFC bölgelerinin örnek şekli.	5
2.2. Çeşitli beyin bölgelerinin mPFC kortikal katmanlarında oluşturduğu sinaptik girdilerin yerleşimi.	7
3.1. eGRASP ve iCre ifadesini sağlayan plazmid şemaları.	18
3.2. Çalışmada kullanılan virüsler ve enjeksiyon bölgeleri.	21
3.3. Çalışmada kullanılan AAV yapıları.	21
3.4. Kronik sosyal yenilgi stres deney takvimi.	22
3.5. KSYS’de kanama bölgesi bağımlı fiziksel stres uygulama algoritması.	25
3.6. Sosyal etkileşim test düzeneği.	26
3.7. Sosyal etkileşim test uygulaması.	27
3.8. Dalga boyu taramalı görüntüleme şeması.	30
3.9. Örnek dalga boyu tarama görüntü sonuçları.	30
3.10. Çalışmada uygulanan çerçeve sıralı görüntüleme yöntemi.	32
3.11. Çerçeve sıralı görüntünün k-ortalama yöntemine ayrıştırılması için kullanılan algoritma.	33
3.12. LUMoS spektral ayrıştırma öncesi ve sonrası örnek sinaptik sinyaller.	33
3.13. Kiremit taramalı görüntüleme şeması.	36
3.14. Çalışmada uygulanan görüntü işleme algoritma sıralaması.	38
3.15. Spektral ayrıştırma öncesi, sonrası ve dekonvolüsyon sonrası örnek sinyaller.	40
3.16. Dekonvolüsyon sonrası dendritik görüntülerin örnek segmentasyonu ve rekonstrüksiyonu.	41
3.17. Segmente dendrit ve dekonvolüe sinaptik sinyallerin yüksek geçirimli filtreleme ile eşikleme.	44
3.18. Sinaptik sinyallerin eşikleme sonrası görüntüleri.	45
3.19. Örnek seçilmiş kortikal kolon bölgelerinde doğrulanmış, otomatik tespit edilen sinaps yüzdeleri.	47
3.20. Otomatik ve elle tespit edilen sinapsların uzaysal ve hacimsel karşılaştırması.	48
3.21. Synapse Detector algoritmasının genel performansının özeti.	49
4.1. Çalışmada incelenen farelerin CD-1 fare sosyal etkileşim bölgesinde bulunmadığı, bulunduğu dönemdeki sosyal etkileşim süreleri ve sosyal etkileşim oranları.	53

4.2. Örnek görüntüde görüntü işleme basamakları.	54
4.3. Synapse Detector algoritmasının örnek sonucu.	55
4.4. 1. katman ve 2/3. katmanda BLA-mPFK sinaplarının tüm tespit edilen sinaplara oranının karşılaştırması.	56
4.5. 5b katmanında ve 6. katmanda BLA-mPFK sinaplarının tüm tespit edilen sinaplara oranının karşılaştırması.	57
4.6. 5b katmanında ve 6. katmanda BLA-mPFK sinaplarının tüm tespit edilen sinaplara oranlarının sosyal etkileşim oranıyla korelasyonları.	57
4.7. 1. katman ve 2/3. katmanda VH-mPFK sinaplarının tüm tespit edilen sinaplara oranının karşılaştırması.	58
4.8. 5b katmanında ve 6. katmanda VH-mPFK sinaplarının tüm tespit edilen sinaplara oranının karşılaştırması.	59
4.9. 5b katmanında ve 6. katmanda VH-mPFK sinaplarının tüm tespit edilen sinaplara oranlarının sosyal etkileşim oranıyla korelasyonları.	59
4.10. Tüm katmanlarda VTA-mPFK sinaplarının tüm tespit edilen sinaplara oranının karşılaştırması.	60
4.11. 5b katmanında ve 6. katmanda VTA-mPFK sinaplarının tüm tespit edilen sinaplara oranlarının sosyal etkileşim oranıyla korelasyonları.	61
4.12. BLA-mPFK, VH-mPFK ve VTA-mPFK sinaplarının kortikal kolon boyu farklı kortikal katmanlardaki oranlarının TBA ile iki boyuta indirgenmiş koordinatları.	62
4.13. 5b katmanı ve 6. katmanda sadece BLA'dan sinaptik girdi alan dendritik segment oranlarının karşılaştırması.	63
4.14. 5b katmanı ve 6. katmanda sadece VH'den sinaptik girdi alan dendritik segment oranlarının karşılaştırması.	64
4.15. 5b katmanı ve 6. katmanda sadece VTA'dan sinaptik girdi alan dendritik segment oranlarının karşılaştırması.	64
4.16. 5b katmanı ve 6. katmanda birden fazla beyin bölgesinden sinaptik girdi alan dendritik segmentlerin oranlarının karşılaştırılması.	65
4.17. 5b katmanında birden fazla beyin bölgesinden sinaptik girdi alan dendritik segmentlerin oranlarının her bir segment alt tipi için karşılaştırılması.	66
4.18. 6. katmanda birden fazla beyin bölgesinden sinaptik girdi alan dendritik segmentlerin oranlarının her bir segment alt tipi için karşılaştırılması.	67
4.19. 5b ve 6. katmanda VH-mPFK sinaplarına en yakın VH-mPFK sinaplarının ve BLA-mPFK sinaplarına en yakın BLA-mPFK sinaplarının tüm gruplar için histogram dağılımları.	69
4.20. 5b ve 6. katmanda VH-mPFK sinaplarına en yakın BLA-mPFK sinaplarının ve BLA-mPFK sinaplarına en yakın VH-mPFK sinaplarının tüm gruplar için histogram dağılımları.	70



<b>4.21.</b> 5b ve 6. katmandaki BLA-mPFK ve VH-mPFK sinapslarının ilk dört BLA-mPFK ve ilk dört VH mPFK sinapslarına en yakın Öklid mesafelerine göre rastgele orman sınıflamasıyla stres duyarlılık sınıflaması.	<b>73</b>
<b>4.22.</b> ROS'nin doğrulama yüzdesinin elde edilen ve davranış fenotip etiketleri rastgele dağılan veriler için karşılaştırılması.	<b>74</b>
<b>4.23.</b> ROS F1 skorlarının elde edilen ve davranış fenotip etiketleri rastgele dağılan veriler için karşılaştırılması.	<b>74</b>
<b>4.24.</b> 5b ve 6. katmandaki BLA-mPFK ve VH-mPFK sinapslarının ilk dört BLA-mPFK ve ilk dört VH mPFK sinapslarına en yakın Öklid mesafelerinin UMAP projeksiyonu.	<b>75</b>
<b>5.1.</b> Çalışmanın özet şekli.	<b>79</b>

## TABLOLAR

Tablo	Sayfa
2.1. Optogenetik veya kemogenetik kullanılarak mPFC nöronlarını hedefleyerek depresyon benzeri davranışları inceleyen çalışmaların özeti.	9
3.1. İntrakraniyel enjeksiyon koordinatları (bregmaya göre).	20
3.2. Çalışmada kullanılan virüsler, derişimler, enjeksiyon bölgeleri, virüslerin protein ifadesini sağladığı nöron alt tipleri.	20
3.3. eGRASP proteinleri ve TagRFP-T floresan proteininin 40x ve 63x yağlı objektifler için Rayleigh çözünürlük sınırları	35
4.1. VH-mPFC sinapsına en yakın VH-mPFC sinapsı ve BLA-mPFC sinapsına en yakın BLA-mPFC sinaps mesafelerinin gruplar arasındaki istatistiksel karşılaştırma sonuçları (Bkz. Şekil 4.19.).	71
4.2. VH-mPFC sinapsına en yakın BLA-mPFC sinapsı ve BLA-mPFC sinapsına en yakın VH-mPFC sinaps mesafelerinin gruplar arasındaki istatistiksel karşılaştırma sonuçları (Bkz. Şekil 4.20.).	72

## 1. GİRİŞ

Stres, ani veya algılanan değişikliklere karşı vücudun iç dengesini sağlamaya yönelik önemli bir yanıttır. Ancak bazı kişilerde, stres yanıtı bozulabilir ve depresyon, anksiyete bozuklukları ve post travmatik stres bozukluğu gibi sebebi bilinmeyen hastalıklar gelişebilir. Bu sebeple, strese duyarlılığın mekanizmasının anlaşılması, stresle ilişkili psikiyatrik hastalıkların önlenmesi ve erken müdahalesi için yöntemlerin geliştirilmesi açısından önem arz etmektedir (1).

Medial prefrontal korteks (mPFC), stres tepkisini düzenleyen bir önemli bölgelerden birisidir. Kronik stres, mPFC 2/3. ve 5. katmanlarında piramidal nöronlarının apikal dendritlerinde ve dendritik çıkıntılarında (*dendritic spine*) atrofiye neden olduğu, bazal dendritlerde değişiklik oluşturmadığı gözlenmiştir (2-7). Kronik stresin bu etkileri devre spesifik olduğu düşünülmektedir: entorhinal kortekse uzanan mPFC piramidal nöron dendritleri kronik stres sonrasında dendritik atrofi gösterirken, bazolateral amigdala (BLA)'ya uzanan piramidal nöronlar dendritlerinin atrofiye direnç gösterdiği gözlenmiştir (8). Ancak, dendritik atrofide gözlenen bu direncin mPFC'ye uzanan piramidal nöronlarda da geçerli olup olmadığı henüz bilinmemektedir.

Kronik stresle ilişkili dendritik atrofi sonrasında, bu dendritlerin üzerindeki sinaptik bağlantıların da kaybolması beklenir. Farklı beyin bölgelerinden mPFK'ye uzanan projeksiyonların sinaptik kayıpları farklı şekilde oluşursa, farklı beyin bölgelerinden gelen sinaptik girdilerin mPFK üzerindeki göreceli ağırlığı değişecek ve dolayısıyla stres yanıtında rol oynayan beyin devresini değiştirecektir. Ancak, şimdiye kadar hangi mPFK sinaptik girdilerinin, dendritik gerileme veya dendritik çıkıntı kaybından etkilendiği bilinmemektedir. Benzer şekilde, önceki çalışmalarda stres duyarlılığı değerlendirilmediği için mPFK dendritlerinde bildirilen morfolojik değişikliklerin stres yanıtında adaptif veya maladaptif etkisinin olup olmadığı da bilinmemektedir.

mPFK, stres tepkisinin düzenlenmesinde önemli olduğu bildirilmiş ventral hipokampus (VH), bazolateral amigdala (BLA) ve ventral tegmental alandan (VTA) sinaptik girdi alır (9, 10). Optogenetik veya kemogenetik yolla mPFK veya mPFK'ye uzanan olan VTA, VH ve BLA nöronlarının aktivasyonunun depresyon ve anksiyete benzeri davranışları değiştirdiği bildirilmiştir (11-22). Duyusal kortekslerin aksine, mPFK agranülerdir, talamustan girdi alan 4. katmanı yoktur ve tüm katmanlardan girdi alır (10). Farklı uzun menzilli girdilerden gelen bilgiler, aynı katman içindeki yerel devreler (*intralaminar*) veya farklı katmanlar arasında (*interlaminar*) toplanarak, sürekli değişen koşullara uyum sağlamak için optimal bir yanıt oluşturulması sağlanır. Ancak önceki çalışmalar genellikle tek katmana odaklandığı için kronik stresin mPFK devrelerini kortikal laminalar boyunca nasıl değiştirdiğine dair az bilgi mevcuttur (23). BLA nöronlarının mPFK 2/3. katmanını; VH nöronlarının mPFK 5. katmanını hedeflemesi uzun menzilli sinaptik girdilerin yerel devreleri seçici şekilde düzenlediğini düşündürmektedir (10, 24-26).

BLA'dan (korku veya korku karşıtı ilişkili) ve VH'den (yaklaşma ve kaçınma nöronları) mPFK uzanan farklı nöron popülasyonları, stres tepkisini ve ilişkili korku yanıtını zıt yönlerde doğru düzenler; mPFK'ye gelen algısal, duygusal ve mekansal verileri entegre ederek korku yanıtının oluşumunu teşvik eder veya korku yanıtını baskırlar (27, 28). Bu zıt işlevlere sahip nöronların mPFK'nin aynı katmanlarına mı yoksa farklı katmanlarına mı projeksiyon yaptığı bilinmemektedir. Bu nedenle, kronik stresin neden olduğu belirli sinaptik girdilerdeki değişikliklerin tüm mPFK katmanlarında incelenmesi, stresin beyin devrelerini nasıl değiştirdiğini ve adaptif yanıtlar ürettiğini

anlamak için kritiktir ve stresle ilişkili ruhsal bozukluklarda nelerin yanlış gidebileceği de anlaşılmasına katkı sağlayabilir.

Bu tez çalışmasında e-GRASP (*enhanced green fluorescent protein reconstitution across synaptic partners*) yöntemiyle mPFK piramidal nöron dendritlerindeki VTA, VH ve BLA glutamaterjik nöronları kaynaklı sinapslar, kortikal katmanlar boyunca haritalanmıştır (29). Önce, stres dirençli (SDir) ve stres duyarlı (SDuy) fareler 10 günlük kronik sosyal yenilgi stresi (KSYS) uygulaması sonrasında yapılan sosyal etkileşim testiyle belirlendi. Ardından stres naif (SN), SDir ve SDuy farelerin mPFK katmanlarını kapsayacak şekilde, yaklaşık  $200 \times 1500 \times 30 \mu\text{m}^3$ 'lük bir hacimden çok spektrumlu, yüksek çözünürlüklü ve 3 boyutlu görüntüler alındı. Görüntü birleştirme, spektral ayrıştırma ve dekonvolüsyon ve rekonstrüksiyon işlemlerinden sonra sinapslar otomatik şekilde tespit edildi.

Tez çalışmasının sonucunda erkek farelerde kronik stres maruziyetiyle mPFK 2/3. ve 5. katman yüzeyinde BLA-mPFK sinapslarının görelî miktarının arttığı; VH-mPFK sinapslarının görelî miktarının azaldığı gözlenmiştir. Ancak 5. katman derini ve 6. katmanda sadece strese duyarlı farelerde BLA-mPFK sinapslarının görelî miktarının arttığı; VH-mPFK sinapslarının görelî miktarının azaldığı gözlenmiştir. Bununla birlikte 5. katman derini ve 6. katmanda VH-mPFK ve BLA-mPFK sinapslarının uzaysal yerleşimlerinin stres maruziyetiyle değiştiği gözlenmiştir. Dendritik segmentler incelendiğinde 6. katmanda birden fazla beyin bölgesinden kaynaklı sinaps bulunduran dentritik segmentlerin görelî miktarının arttığı gözlenmiştir. Bu dentritik segmentler incelendiğinde VTA kaynaklı sinapsların BLA veya VH kaynaklı sinapslarla varlığının stres duyarlılığının kestiriminde etkili olduğu gözlenmiştir. 5. katman derini ve 6. katman BLA ve VH kaynaklı sinapsların birbirlerine olan uzaysal konumları incelendiğinde strese duyarlı farelerde VH sinapslarına en yakın BLA sinapsın Öklid mesafesinin azaldığı ancak BLA sinapslarına en yakın VH sinapsın Öklid mesafesinin arttığı gözlenmiştir.

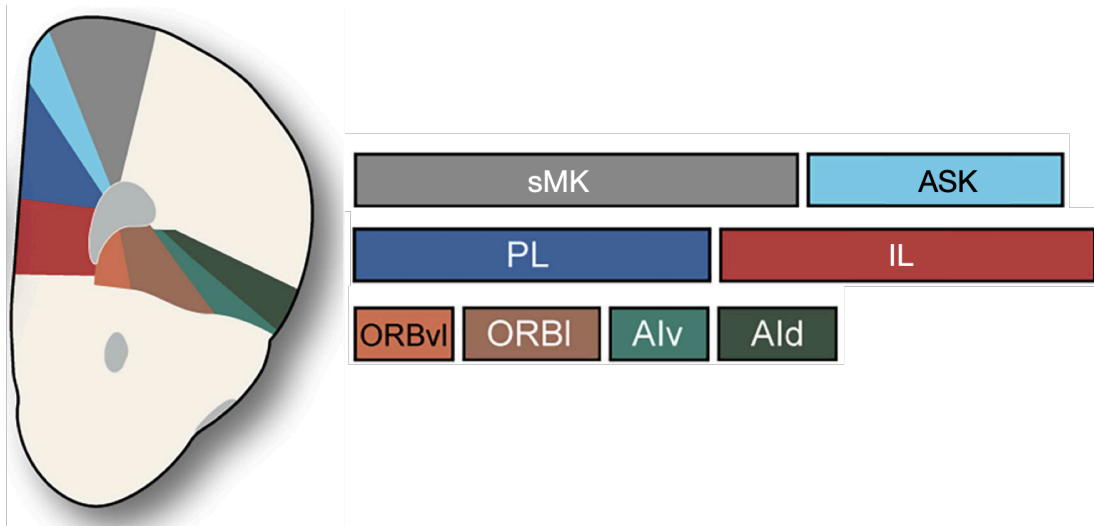
Bu tez çalışmasındaki bulgular stres duyarlılığında hem beyin bölgesi hem de katman spesifik devrelerin kortikal katmanlar boyunca etkili olduğu ve derin beyin

katmanlarındaki sinaptik yeniden düzenlemenin stres duyarlılığında ve strese adaptif yanıtın oluşmasında etkili olduğunu düşündürmektedir.

## 2. GENEL BİLGİLER

### 2.1. Medyal Prefrontal Korteks Anatomisi ve Devreleri

Medyal prefrontal korteks (mPFK) beynin ön medyal bölgesinde bulunan beyin bölgesidir. PFK'nin medyal bölgesinde yer alır (30). Hem anatomik incelemelere göre hem de PFK bağlantılarının incelediği çalışmalar göz önüne alındığında mPFK'nin dorsal (dmPFK) ve ventral (vmPFK) bölgelerine ayrıldığı düşünülmektedir. Daha detaylı bir sınıflamada da dmPFK'nin sekonder motor korteks (sMK) ve anterior singulat kortekse (ASK); vmPFK'nin de prelimbik (PL), infralimbik (IL) ve medyal orbitofrontal korteks (mORB) bölgelerine ayrıldığı tariflenmiştir (31) (Şekil 2.1.).



**Şekil 2.1.** Fare mPFK bölgelerinin örnek şekli.

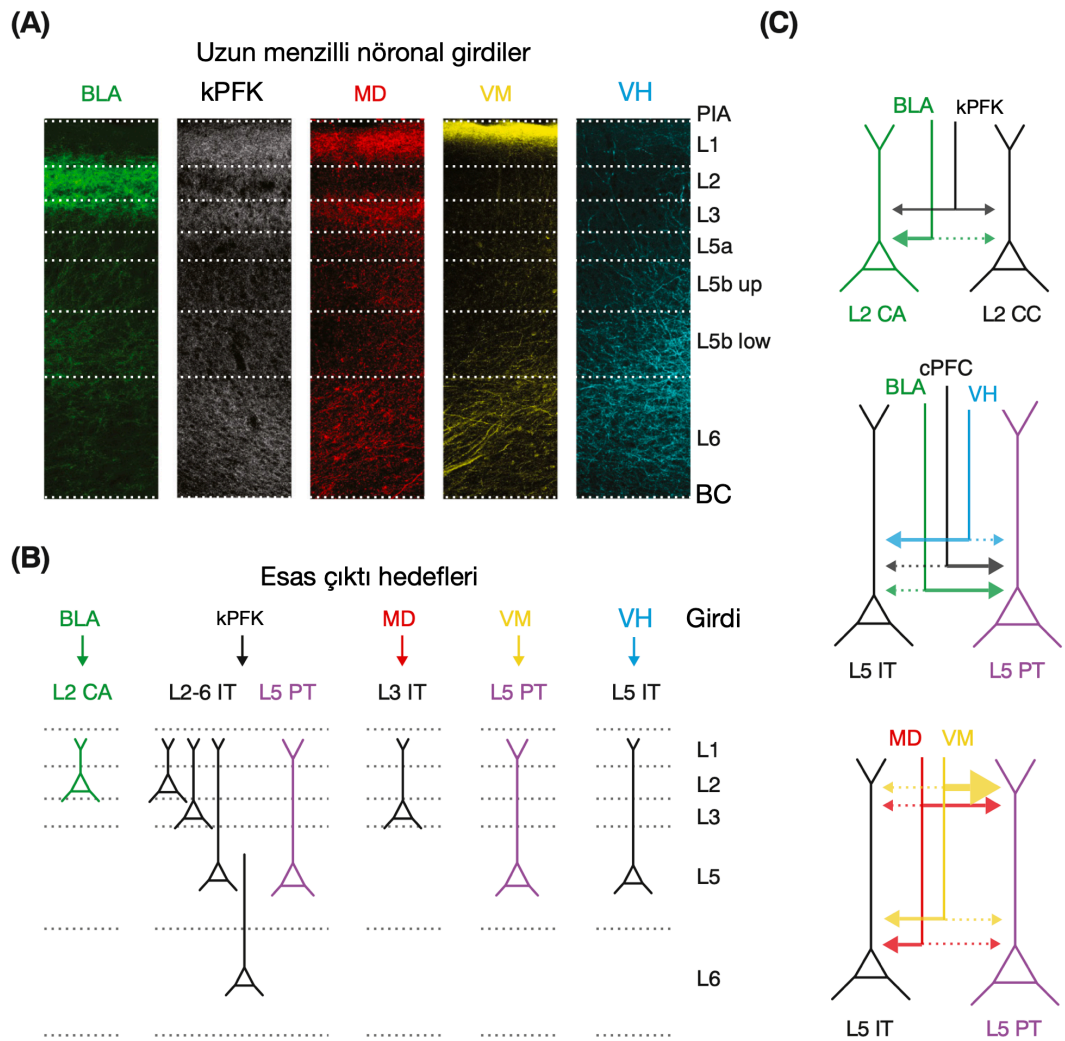
Kortikal bölge isimleri korteks içindeki renkler ve kortekste kapladıkları alana göre ayarlanarak korteks yanında gösterilmiştir. Üst satır dmPFK'ye; orta satır vmPFK'ye ve son satır ventrolateral PFK (vIPFK) ait beyin bölgelerini ifade etmektedir. mORB şekilde görünmemektedir. Le Merre ve ark.(31)'dan alınmıştır. Kısaltmalar: sMK: sekonder motor korteks, ASK: anterior singulat korteks, PL: prelimbik korteks, IL: infralimbik korteks, ORBvl: ventrolateral orbitofrontal korteks, ORBI: lateral orbitofrontal korteks, Alv: ventral agranular insular korteks, Ald: dorsal agranular insular korteks

Beyinde duyu korteksi başta olmak üzere korteksin geneli 6 kortikal katmana ayrılır. 1. katmanda genellikle piramidal nöronların dendritleri ve belirli internöron popülasyonları bulunmaktadır (10, 32). 2. ve 3. katman genellikle 2/3. katman şeklinde birlikte gruplanmaktadır ve çeşitli kortikal alanlara uzanan (intertelensefalik (IT))

nöron somalarını barındırır (33). 4. katman genellikle talamus kaynaklı girdilerin bulunduğu bölgedir ve diğer kortikal katmandaki nöronlarla bölgesel bağlantılar oluşturur (34, 35). 5. katmanın da kortikal çıktı katmanı olduğu kabul edilir. Çeşitli kortikal bölgelere uzanan IT nöronlara ek subkortikal beyin bölgelerine uzanan piramidal yol (*pyramidal tract, PT*) nöronlarını da içerir (36). Ancak talamik girdilerin duyu kortekslerinin derin katmanlarına da sinaptik girdi sağladığını gösteren veriler de bulunmaktadır (37). Son olarak 6. katman da 5. katman gibi bir çıktı katmanıdır ve talamusa uzanan nöronları (Kortikotalamik, CT) bulundurur (36).

Ancak mPFK katmanları diğer kortikal bölgelerden farklılık göstermektedir. Öncelikle, mPFK agranüler bir kortektir ve 4. katmanı içermez. Katman sıralamasında 2/3. katmandan sonra 5. katman gelir (10, 31, 38-41). İkincisi mPFK çıktı nöronları 2. katmandan 6. katmana kadar yaygın şekilde bulunabilir ve bu çıktılar hem lokal nöronlara da uzanabilir. Sonrasında bu nöronlar kendi aralarında bölgesel devreler kurarak mPFK çıktı oluşumuna katkıda bulunabilir (10). Son olarak subkortikal beyin bölgeleri mPFK'nin bütün katmanlarına sinaptik girdi sağlayabilir. Ancak sinaptik girdiler katman, bölge ve nöron spesifik şekilde düzenlenir (10, 24, 25, 42) (Şekil 2.2.).

Optogenetik ve channelrhodopsin yardımcı devre haritalama (*Channelrhodopsin assisted circuit mapping, CRACM*) yöntemiyle bir beyin bölgesinden kaynaklı sinaps sonrası piramidal nöron dendritleri üzerindeki girdilerin dağılımını elektrofizyolojik yolla görebilmek mümkündür (24, 43-46). Bu incelemelerde BLA'dan kaynaklı mPFK sinaptik girdilerin 2. ve 3. katmanlarda daha fazla bulunduğunu; VH kaynaklı sinaptik girdilerin 2/3.ve 5. katmanda yaygın şekilde bulunduğunu gözlenmiştir (24, 26). Bu çalışmalarda apikal ve bazal dendrit ayrımı net şekilde yapılmasa da bildirilen koordinatlar BLA kaynaklı sinaptik girdilerin mPFK 2/3. katmanlarındaki nöronlarının bazal dendritlerinde; VH kaynaklı sinaptik girdilerin mPFK 5. katmanındaki nöronlarının apikal dendritlerinde daha fazla bulunduğunu düşündürmektedir (10, 24, 26) (Şekil 2.2.). Ayrıca VH kaynaklı girdilerin mPFK içerisinde uzaysal açıdan VH'nin yüzeyinden ve derininden projekte olanlar olmak üzere 2 farklı gruba ayrıldığı, bu nöronların elektrofizyolojik özelliklerinin farklı olduğu görülmüştür (28).



**Şekil 2.2.** Çeşitli beyin bölgelerinin mPFC kortikal katmanlarında oluşturduğu sinaptik girdilerin yerleşimi.

A) Tüm kortikal katmanlar boyunca yapılan sinaps öncesi akson sonlanması görüntülemelerinin örnek resimleri, B) BLA, kontralateral PFK (kPFC), medyodorsal talamus (MD) ventromedyal talamus (VM) ve VH'nin oluşturduğu sinaptik girdilerin dağılımı ve C) Beyin bölgelerinin mPFC'deki seçici IT ve PT nöron bağlantıları gösterilmiştir. BLA nöronları mPFC 2/3. katman nöronlarıyla çift yönlü bağlantı oluşturmaktadır. VH mPFC 5. katman nöronlarındaki IT nöronlarına daha fazla girdi sağlamaktadır. MD ve VM mPFC kortikal katmanlarına yaygın şekilde nöron uzantısı sağlamakta ve bunu seçici şekilde sağlamaktadır. Şekil Anastasiades ve Carter, (10)'dan alınmıştır.

## 2.2. Depresyon ve Nöral Devreler

İnsanlar ömürleri boyunca çok kez stresli olaylarla karşı karşıya kalırlar. Yapılan çalışmalarda bazı insanların stresli olaylarla baş edebildiği, bazılarında ise depresyon, travma sonrası stres bozukluğu gibi psikiyatrik hastalıklar görülebildiği;



diyabet, hipertansiyon vb tıbbi hastalıkların morbiditesinin arttığı bildirilmiştir. Stres maruziyeti sonrasında görülen bu farklılıkların nedeni halen tam anlaşılamamıştır (1).

Normal koşullarda PFK limbik sistemin aktivitesini denetim altında tutar (yukarıdan aşağıya kontrol), stresörle karşılaşıldığında amigdala aktivitesi artar, PFK aktivitesi geçici bir süre için inhibe olur, ancak yeni duruma uygun yanıtlar sağlandıktan sonra amigdala aktivitesi azalır, kognitif işleyişte tekrar yukarıdan aşağıya kontrol hakim hale gelir. Artmış amigdala ve hipotalamopitüiter eksen (HPA) aktivitesi hipokampus ve PFK tarafından kontrol altına alınır. Afektif bozukluklarda (depresyon ve anksiyete bozuklukları) amigdala ve HPA aktivitesindeki artış ile PFK ve hipokampus aktivitesindeki baskılanmanın devam ettiğini düşündüren bulgular vardır (47).

Görüntüleme çalışmalarında depresyon hastalarında hipokampus ve PFK hacminin azaldığı, bazolateral amigdala (BLA) hacminin ise arttığı gözlemlenmiştir (48-50). Fonksiyonel manyetik rezonans görüntüleme (fMRG) çalışmalarında depresyon hastalarında BOLD (*blood-oxygen-level-dependent*) sinyallerinin PFK'de azaldığı, amigdalada ise arttığı gözlemlenmiştir (51). Bunun yanı sıra depresyon hastalarında olumsuz duygulanımla amigdala ve subgenual anterior singulat korteks arasındaki bağlantı gücünün arttığı; ventral hipokampus (VH) ve mPFK arasındaki bağlantı gücünün ise azaldığı görülmüştür (52, 53). Kronik strese maruz kalıp strese yatkınlık gösteren sıçanların MR görüntülerinde ventral tegmental alan (VTA) ve hipokampus hacminde azalma ve fMRG analizlerinde strese yanıt olarak mPFK-hipokampus bağlantı gücünde artma görülmüştür (54). Ancak MRG'nin çözünürlüğünün istenen yükseklikte olmaması, fMRG'de gözlenen aktivite değişikliklerinin hangi nöron gruplarından kaynaklandığının (aktivite artışının eksitator mü, inhibitör mü nöronlardan kaynaklandığı, ilgili bölgenin çıktısını ters yönlere etkiler) anlaşılamaması ve klinik olarak heterojen hasta populasyonlarının incelenmesi, çalışma planlarının farklı olması gibi sebeplerden ötürü görüntüleme çalışmaları strese yatkınlığa yol açan mekanizmalarla ilişkili kısıtlı bilgi vermektedir (55, 56).

Beyin bağlantı haritalarının ortaya çıkarılması aracılığıyla yapılan çalışmalarda farede VH, BLA ve VTA'nın mPFK'ye aksonal uzantı gönderdiği tespit edilmiştir (9, 57). Optogenetik uyartım yöntemi kullanılarak bu bölgelerden birinden diğerine giden

bağlantıları seçici olarak uyararak bölgeler arasındaki bağlantıların işlevleri davranış öncesinde veya sırasında incelenebilir (58) (Tablo 2.1.).

**Tablo 2.1.** Optogenetik veya kemogenetik kullanılarak mPFC nöronlarını hedefleyerek depresyon benzeri davranışları inceleyen çalışmaların özeti.

Sinaps Öncesi Bölge	Kullanılan Hayvan	Yöntem	Hücre alt tipi	Uyarı	Strese duyarlılık	Kaynak
BLA	C57BL/6	Opto.	pir.	akt. inh.	artma azalma	(15)
	Long-Evans Sıçan	Opto.	pir.	akt. inh.	artma azalma	(16)
		Kemo.	ns	inh.	azalma	
	C57BL/6	Opto.	pir.	inh.	azalma	(17)
	C57BL/6	Opto.	pir.	inh.	azalma	(59)
	C57BL/6	Opto.	pir.	akt.	artma	(60)
C57BL/6	Opto.	pir.	inh.	azalma	(61)	
VH	SD sıçan	Kemo.	ns	akt. inh.	azalma artma	(14)
	129SvevTac	Opto.	pir.	inh.	azalma	(21)
	C57BL/6	Kemo.	ns	akt.	artma	(62)
				inh.	azalma	
	129SvevTac	Opto.	pir.	akt.	artma	(22)
	Calb-Cre; C57BL/6	Opto.	ns	akt.	artma	(28)
C57BL/6	Opto.	pir.	akt. inh.	azalma artma	(61)	
VTA	C57BL/6	Opto.	ns	akt. inh.	değişme yok artma	(12)
	TH::IRES-Cre; C57BL/6	Opto.	dop.	akt.	azalma	(13)
	TH::Cre C57BL/6	Opto.	dop.	akt.	azalma	(63)
	TH::Cre C57BL/6	Kemo.	dop.	inh.	artma	(64)

Kısaltmalar: Opto.: Optogenetik, Kemo.: Kemogenetik, pir.: piramidal nöron, dop.: dopaminerjik nöron, ns: non-spesifik, akt.: aktivasyon, inh.: inhibisyon.

Farelerde ve sıçanlarda BLA'daki piramidal nöronlardan mPFK'teki piramidal nöronlara giden bağlantılar optogenetik olarak aktive edildiğinde korku ve anksiyete benzeri davranışların arttığı ve sosyal etkileşimin azaldığı; inhibe edildiğinde ise korku ve anksiyete benzeri davranışlarının azaldığı, sosyal etkileşimin arttığı ve hatta edinilmiş korku yanıtının sönmülenebildiği gözlemlenmiştir (15-17, 59, 60).

Farede VTA'daki nöronların hem piramidal hem de internöronlarla sinaptik bağlantı yapabildiği tespit edilmiştir (65). VTA dopaminerjik nöronlarından mPFK nöronlarına uzanan bağlantıların optogenetik aktivasyonu ile kuyruktan asma testi sırasında daha çok çarpınma davranışı ve strese duyarlı farelerde sosyal etkileşim artışı gözlemlenmiştir (12, 63). Tam aksine bu bağlantılar optogenetik inhibe edildiğinde strese duyarlı farelerde sosyal etkileşimin azaldığı ve açık alan testinde merkezde geçirilen sürenin azaldığı gözlemlenmiştir (13, 64). Bu çalışmalar, VTA-mPFK bağlantısının uzun süredir bilinen VTA – akümbens çekirdek (*Nucleus accumbens*, NAc) arasındaki stres duyarlılığını belirleyici bağlantısından bağımsız bir rolünün olduğunu göstermektedir (13). Ancak VTA kaynaklı dopaminerjik nöronların yarısının aynı zamanda glutamat salıverebilmesi bu devrenin strese duyarlılığın oluşumunda modulator bir etkiye sahip olabileceğini düşündürmektedir (13, 66-68).

Farelerde VH'den mPFK'ye olan piramidal nöron bağlantılarının işlevi ise ikirciklidir: Bir çalışmada bu devrenin optogenetik olarak aktive edilmesiyle anksiyete benzeri davranışların arttığı, optogenetik olarak inhibe edilmesiyle de bu davranışların azaldığı gözlemlenmiştir (21, 22, 28, 62). Ancak başka bir çalışmada sıçanlarda düşük doz ketaminin antidepresan etkisi için VH- mPFK devresinin işlevsel olması gerektiği bildirilmiştir (14). Bu devre optogenetik yolla inhibe edildiğinde ketaminin antidepresan etkisi ortadan kalkmış ve sosyal hafıza oluşumunun önlendiği gözlemlenmiştir (14, 69). Bu ikilem, mPFK piramidal nöron dendritlerinde VH dışında diğer beyin bölgelerinden köken alan ve ketamin tarafından düzenlenen girdilerin yanıtı etkilediğini düşündürmektedir.

### 2.3. Dendritler, Sinaptik Plastisite ve Dendritik Bilgi İşleme

Dendritler nöronların en temel işlemsel birimidir. 19. yy sonlarında Cajal tarafından tanımlanmıştır. Dendrit yapılanması tüm sinir hücresi çeşitlerinde farklı şekillerde görünür. Örneğin beyincikteki Purkinje hücrelerinde uzun ve çok dallı iken; ara nöronlarda kısa ve künt yapılıdır. Bu durum sinir hücrelerinin bilgi işleme faaliyetinin hücre tipleri arasında farklılık göstermesini sağlamaktadır. Dendrite çok basit yapı olarak baktığımızda dendritik çıkıntı (*spine*) ve dendritik gövde (*shaft*) yapıları göze çarpmaktadır. Dendritik çıkıntı yapısında dendrit başı ve dendrit boynu morfolojik olarak tespit edilmiştir. Dendrit başlarının üzerinde sinaps sonrası yoğunluk adı verilen çeşitli düzenleyici protein ve yapıların bulunduğu bir alan vardır. Bu bölgede PSD-95, beta-katenin, nöroligin gibi dendritin yapısını ve işlevini düzenleyen proteinler bulunur (70).

Dendritik çıkıntılar yapısal olarak mantar (*mushroom*), künt (*stubby*), ince (*thin*) ve yeni uzantı (*filopodia*) şeklinde görülebilir. Künt, ince ve yeni uzantı dendritik çıkıntılarının yapısal olarak daha genç, yani yapısal değişimlere daha uygun, ancak mantar yapılı dendritik çıkıntılarının yapısal olarak daha olgun olduğu, yani yapısal değişime daha kapalı veya daha yavaş şekilde düzenlendiği düşünülmektedir (71). Yapılan çalışmalarda doğum sonrası dönemde yeni uzantı yapısındaki dendritik çıkıntılarının daha fazla görüldüğü ve erişkinliğe doğru giderek azaldığı gözlemlenmiştir. Bu durumun tam tersi mantar dendritik çıkıntılar için geçerlidir. Mantar dendritik çıkıntılarının yoğunluğu yaşla birlikte artmaktadır. Ancak dendritik çıkıntılarının yapılanması ve düzenlenmesi işlenen bilgiye ve bilginin dendrit üzerinde işlendiği bölgeye bağımlı olarak değişmektedir (72, 73).

Dendritik çıkıntılarının oluşumu sonrasında işlevsel sinaptik bağlantı oluşturması canlının yaşamsal dönemine ve incelenen beyin bölgesine bağlı olarak değişmektedir. Yapılan çalışmalarda erişkin primer duyu korteksinde dendritik çıkıntı oluştuktan 4 gün sonra presinaptik bölgedeki aksonlarla bağlantı oluşturduğu görülmüşken; hipokampal bölgede dendritin sinaps oluşturmasının 1 gün kadar sürdüğü; doğum sonrası erken dönem beyin dilimlerinde bu sürecin 1 saatten daha kısa sürebildiği görülmüştür (71, 74, 75).

Beyin başlangıçta sanıldığı gibi gelişim döneminden sonra statik kalan bir yapı değildir, aksine dinamikdir, değişen çevresel koşullara göre yeni sinapsların oluştuğu ve var olanların budandığı gösterilmiştir. Sinaptik plastisite, çeşitli etkilerle sinaps öncesi ve sonrası nöronların bağlantı gücünün değişmesidir (76). Sinaptik plastisite esnasında dendrit ve aksonlar arasında yeni bağlantılar oluşabilir ve kurulan bağlantılar bozulabilir. Bu bağlantıların oluşumu ve bozulması uyarının biçimine (frekans, salıverilen nörotransmitter miktarı vs.) ve uyarı zamanlamasına (sinaps öncesi ve sonrası nöronların hangisinin daha önce uyarıldığına bağlı olarak) göre değişebilir (77).

Sinaptik bağlantının güçlenmesi o bölgede uzun süreli güçlendirme ile (*LTP = long term potentiation*) ilişkilirken sinaptik bağlantının zayıflaması uzun süreli depresyon (*LTD = Long Term Depression*) ile ilişkilidir. Bir dendrit dalı üzerinde elektriksel veya biyokimyasal yolla LTP veya LTD indüklenmesi çevresindeki dendritlerin LTP veya LTD oluşumunu etkiler. LTP veya LTD indüksiyonu dendrite yapısal değişiklik olarak yansıyabilir (17, 77-79).

Elektrofizyolojik çalışmalar sonucunda, sinaps sonrası dendritlerin sinaps öncesi aksonlardan kaynaklı girdileri çeşitli mantıksal işlemler ve filtrelerden geçirdikten sonra somada toplandığını ve bu toplamın somada değerlendirilmesiyle somada aksiyon potansiyeli oluşup oluşmamasına karar verildiği öne sürülmüştür (72, 80-84). İki dallanma noktası arasındaki dendrit dallarının da sinaps öncesi nöron dendritlerindeki bu işlemlerin temel birimi olduğu hipotez edilmiştir (72, 85-89). Örneğin, sıçan hipokampus dilimlerinde yapılan incelemelerde, tek dendrit üzerine uygulanan glutamatın serbestlenmesi (*uncaging*) sonucu oluşan kalsiyum dalgasının dendritik çıkıntıdan en fazla 10 µm uzağa yayılabildiği görülmüştür (90). Benzer şekilde Ras, RhoA ve Rac dendritik proteinlerin de yayılımının 10 µm ile sınırlı olduğu bulunmuştur (85, 91). Ayrıca, distalde dallanma yapan dendritlerin dallanma noktasındaki potasyum kanalı yoğunluğunun artışı sebebiyle somadan ölçülen potansiyelin kaybı, iki dallanma bölgesi arasındaki dendrit segmentinin dendritik lokal bilgi işleme birimi olduğu düşüncesini güçlendirmektedir (92, 93).

Dendrit dalları kendisine gelen bilgileri işlerken biyokimyasal, aktif ve pasif elektrofizyolojik işlemeye kadar birçok yöntemi kullanabilir. Bu işleme yöntemleri sinapsların aktive olma şekline, aktive olan sinapsların dendrit üzerindeki

yoğunluğundan etkilenmektedir (82, 89, 93-95). Nöron kültürlerinde dendrit dalı üzerindeki sinaps uyarım yönü somaya doğru olacak şekilde ayarlandığında aksiyon potansiyeli oluşum ihtimalinin arttığı gözlenmiştir (93). Ayrıca bazı dendrit dallarının sinaptik kümelenmeye (*cluster sensitive*) veya dağınıklığa (*scatter sensitive*) hassas olduğu tespit edilmiştir. Sinaptik kümelenmeye hassas olan nöronların sadece bir dendritinde eş zamanlı uyarımların aksiyon potansiyeli oluşturma ihtimali yüksekken; sinaptik dağınıklığa duyarlı olan nöronların aksiyon potansiyeli oluşturma için birden fazla dendrite eş zamanlı uyarımın ulaşması gerekmektedir (94).

Yapılmış çalışmalarda sinaptik kümelenmenin genellikle primer duyu kortekslerinde duyuusal bileşenin temsil kapasitesinin artırılmasında etkili olduğu gösterilmiştir (96). Sinaptik dağınıklığın da genellikle asosiyasyon kortekslerinde çoklu duyuusal girdilerin toplanmasında etkili olduğu gözlenmiştir (97).

#### 2.4. Medyal Prefrontal Kortekste Dendritik İşleme

mPFK nöronlarının dendritik işleme mekanizmaları diğer kortikal bölgelere göre daha az bilinmektedir. Ayrıca diğer beyin bölgelerinden kaynaklı aksonal uzantıların korteksin hangi tabakasında sonlandığı bilinmesine rağmen, bu sonlanmaların nöron somasına göre nerede olduğu halen bilinmemektedir. Buna rağmen in vitro elektrofizyolojik kesitlerde mPFK dendritlerinin nörotransmitter ve nöromodulatörler reseptör aktivasyonlarıyla elektrik aktivite değişimlerinin incelendiği çalışmalar bulunmaktadır (98-104).

mPFK 5. katman piramidal nöronların dendritik aksiyon potansiyelleri (*dendritic spike*) duyu korteksinden daha farklı şekilde gerçekleştiği gözlenmiştir. Dendritik aksiyon potansiyelleri somatosensoriyel kortekste ağırlıklı olarak kalsiyum akımlarıyla gerçekleşirken, mPFK'de sodyum akımıyla gerçekleştirilmektedir (98, 99). Ayrıca mPFK piramidal nöronlarının bazal dendritlerinin dallanma noktası, dal ve dendrit dalının en uç nokta sayısı ile piramidal nöronun uyarılabilirliği arasında doğru orantılı korelasyon tespit edilmiştir (100).

Dembrow ve arkadaşları (101), hipokampal girdilerin PT nöronlarından ziyade IT nöronlarla selektif şekilde sinaps yaptığını gözlemlemişler. PT nöronların dar bir zaman penceresi içerisinde birçok dendritteki uyarımla aktive oldukları ancak

hipokampusla sinaps yapan IT nöronları daha geniş bir zaman penceresinde gama bantındaki uyarılara daha iyi yanıt verdikleri gözlenmiştir. Zaman pencerelerinde görülen bu farklılık PT nöronların rastlantısal girdileri algıladığını (*coincidence detection*); IT nöronların zamana bağlı girdileri toplayabildiğini düşündürmektedir (105).

Kalmbach ve arkadaşları grup 1 metabotropik glutamat reseptörlerin (mGluR) aktivasyonu ile mPFK 5. katmanındaki PT nöronların proksimal dendritlerindeki hiperpolarizasyonla aktive katyon kanallarının (HCN) aktivitesini azaltarak yavaş depolarizasyon (*slow after depolarization*) aktivite amplitüdünü arttırdığını gözlemlediler. Bu değişiklik piramidal nöronlarının elektrik aktivitesinin daha uzun süreli olmasını sağlayabilir.

Seong ve Carter (103) ve Gee ve arkadaşları (104) da D1R ve D2R dopamin reseptör alt tiplerinin ifade eden piramidal nöronların morfolojik ve elektrofizyolojik açıdan farklı gruplara ait olduklarını tespit etmişler. Seong ve Carter (103), D1R pozitif piramidal nöronların çok az düzeyde HCN kanal aktivitesine sahip olduğunu gösterirken; Gee ve arkadaşları (104) D2R pozitif piramidal nöronların PT alt tipinde olduğunu, HCN kanal aktivitesi gösterdiğini ve D2R aktivitesiyle uyarılabilirliğinin arttığını gözlemlemişler.

Wang ve arkadaşları (106)  $\alpha 2$  adrenerjik reseptörlerle ( $\alpha 2$ -AR) ve HCN kanallarının aynı dentritik çıkıntı üzerinde ifade edildiklerini gözlemlediler. Yine aynı çalışmada  $\alpha 2$ -AR aktivasyonu ile dentritik çıkıntı içerisinde siklik AMP üretiminin azaldığını, ardından HCN kanallarının etkinliğinin azaldığını gözlemlemişler. Bu etkinin de maymunlarda okulomotor uzaysal gecikme yanıtı testi sırasında gecikme ilişkili PFK'deki nöronlarının açılma seçilimlerinin artmasıyla ve sıçanlarda da alterne T labirent testinde daha yüksek oranda doğru seçim oranının artışıyla ilişkilendirmişler.

Hem nörotransmitter hem de nöromodulatörler mPFK nöronlarının aktivitesini çeşitli reseptörler aracılığıyla dentritik düzeyde düzenlemektedir. Ancak bu reseptörlerle presinaptik girdilerin dentritik düzeydeki dağılımı arasındaki ilişki halen bilinmemektedir.

## 2.5. Depresyon ve Dendritler

Majör depresyon hastalarının ölüm ardı beyin incelemelerinde, prelinik çalışmalarda birçok kronik stres davranış modellerinde mPFC 2/3. ve 5. katman piramidal nöron dendrit şeklinde bozulma, dendritik çıkıntı kaybı, dendrit uzunluğunda azalma gözlenmiştir (2-4, 6, 107-109).

Kronik stres sonrası dendritlerdeki bu morfolojik değişimlerin nedeni halen bilinmemektedir. Prelinik modeller anti depresan tedaviler sonrasında kronik stresle ilişkili morfolojik değişikliklerin ortadan kalktığını göstermektedir (110-112). *In vivo* görüntüleme çalışmaları hızlı etkili antidepresanların ara nöronların disinhibisyonu üzerinden dendritik çıkıntı oluşumunu ve kalıcılığını artırdığını göstermektedir (7, 113, 114). Burada ketamin ve psilocibinin en nihayetinde piramidal nöron dendritlerinin bölgesel uyarılabilirliğini değiştirmesi yoluyla dendritik çıkıntının kalıcılığı ve uzamış antidepresan etkinin sağlandığı hipotez edilmiştir (115). Moda-Sava ve arkadaşları (7) farelerde multifoton altında farklı günlerde alınan mükerrer görüntülerde kronik kortikosteroid uygulamasının depresyon benzeri yanıt ve dendritik çıkıntı kaybına yol açtığını, ardından uygulanan ketamin tedavisi sonrası oluşan dendritik çıkıntıların kaybolan dendritik çıkıntılara ~3 µm uzaklıkta yeniden oluştuğunu gözlemlediler ve bu oluşumun antidepresan davranışla ilişkisini gösterdiler.

Kronik stresle görülen bu değişiklikler beyin bölgesi ve kortikal katman spesifik olabilir (8, 116, 117). Shansky ve arkadaşları 10 gün hareketsiz bırakılan erkek sıçanların beyinlerinde entorinal kortekse uzanan piramidal nöron dendritlerinde atrofi gözlemlemişler ancak BLA'ya uzanan mPFC piramidal nöron dendritlerinde herhangi bir değişiklik gözlemlememişler. Liu ve arkadaşları da (116) farelere 10 gün boyunca hareket kısıtlama uyguladıktan sadece dmPFC'den BLA'ya tek yönde uzanan nöronların aktivitesinin arttığını ancak BLA'ya resiprokal uzanan nöronların aktivitesinde bir değişiklik olmadığını gözlemlemişler. Sonraki çalışmalarında kronik hareket kısıtlılığı sonrasında gözlemledikleri dmPFC'deki aktivite artışının sadece 5. katmanda BLA'ya uzanan piramidal nöronlarda olduğunu ancak 2/3. katmandaki piramidal nöronlarda olmadığını gözlemlemişler. Shansky ve arkadaşlarının (8) bulgusuna benzer şekilde dmPFC'den NAK'ye uzanan piramidal nöronların aktivitesinde de herhangi bir değişiklik saptamamışlar (117). Bu bulgular mPFC'de kronik stresle beraber



mPFK'den uzanan piramidal nöronların dendritlerinde girdi ve çıktı spesifik değişikliklerin olduğunu düşündürmektedir.

### 3. GEREÇ VE YÖNTEM

Bu projedeki tüm deney süreçleri Hacettepe Üniversitesi Hayvan Yerel Etik Kurulu tarafından 2019/78 numarasıyla onaylanmıştır.

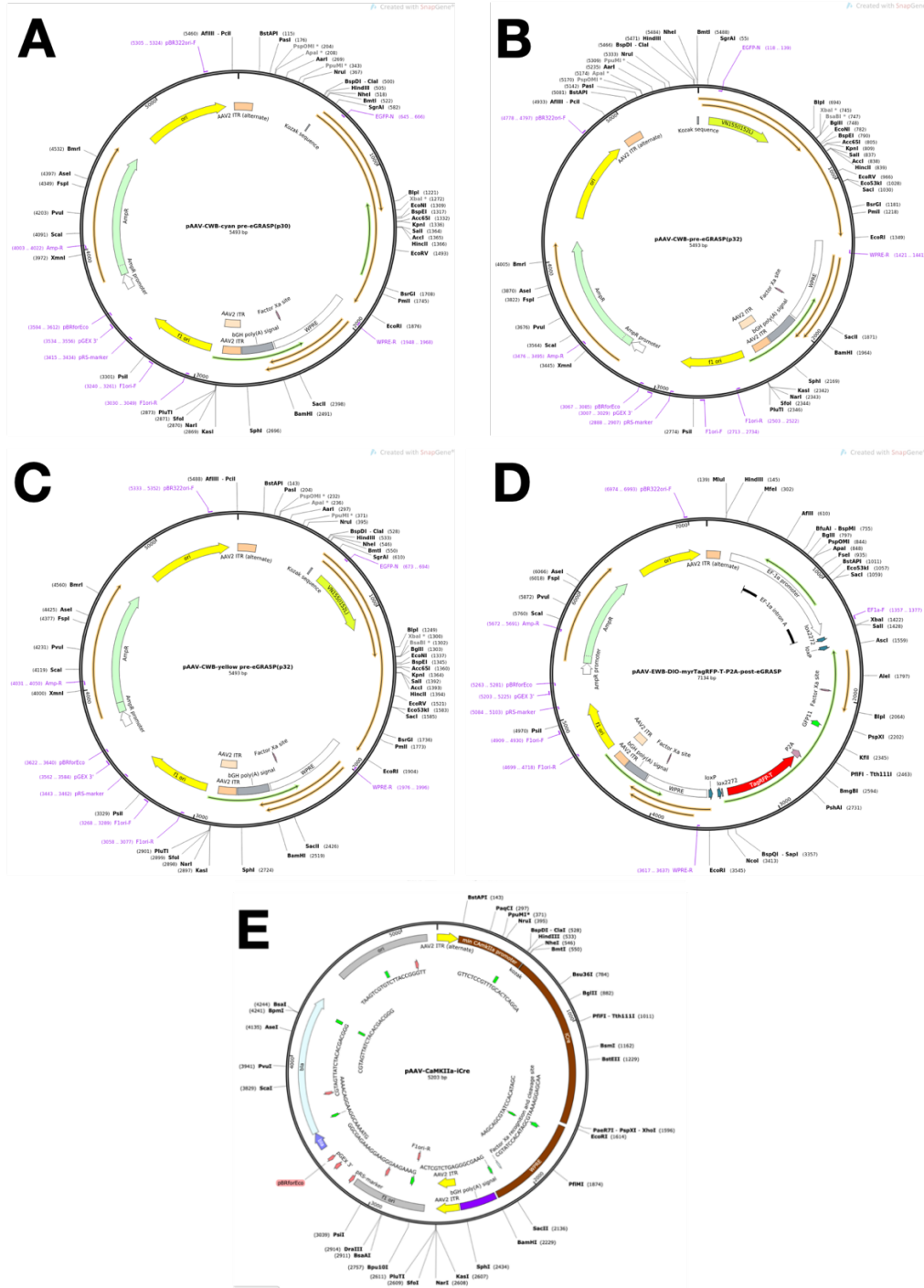
#### 3.1. Fareler

Çalışmada, 12-16 haftalık erkek C57BL/6 cinsi ve 4-24 aylık damızlıktan kesilmiş (*retired breeder*) erkek CD-1 cinsi fareler kullanıldı. Deney süresince farelerin refahı için gerekli önlemler alındı. Fareler laboratuvara getirilmelerinden itibaren deney sürecine kadar  $22 \pm 3$  °C'de 12 saat/12 saat: aydınlık/karanlık ortam sağlayan bir odada, uygun büyüklükteki kafeslerin içinde 4-6'lı gruplar halinde barındırıldı. Tüm deney süreçlerinde farelerin *ad libitum* şekilde beslenmesi sağlandı.

Kronik strese maruz bırakılan fareler, virüs enjeksiyonlarından sonra kafeslerinde tek başına bekletildi. Stres naif grup fareler enjeksiyondan 5-10 gün sonra arasında pleksiglas ayraç olacak şekilde başka bir fareyle etkileşmesi sağlandı (118).

#### 3.2. Virüsler

Dr. Bong Kiun Kaang tarafından Addgene'e bağışlanan, kalsiyum kalmodulin 2 kinaz alfa (CaMKII $\alpha$ ) promoterine bağımlı camgöbeği (#111586), yeşil (#111598) ve sarı (#111580) pre-eGRASP ve Cre rekombinaz enzimine bağımlı post-eGRASP ve TagRFP-T (#111581) proteinlerinin ifadelerini sağlayan plazmidler Addgene şirketinden katı agar içindeki *E.coli* içerisinde temin edildi. Plazmidi içeren *E.coli* ampisilinli katı agar içeren petri kabına yayılıp çoğaltıldıktan sonra yaymadaki tek koloni ampisilinli sıvı agarda çoğaltıldı. Plazmid saflaştırma kitleri (Qiagen) kullanılarak *E.coli* içindeki plazmid dışarı çıkarıldı ve saflaştırıldı. Üretilen plazmidlerin sekansları Sanger sekanslama ile doğrulandı. Saflaştırılan plazmidler adeno-asosiyasyon virüsü (AAV) içerisine paketlenmek üzere Berlin Üniversitesi Tıp Fakültesine bağlı Charite Vektör Merkezine gönderildi. Bu merkezde plazmidler AAV2/1 içerisine paketlenildi. Ayrıca bu merkezden CaMKII $\alpha$  promoterine bağımlı iCre rekombinaz enziminin ifadesini sağlayan AAV de AAV2/1 içerisinde paketlenmiş şekilde temin edildi (Şekil 3.1.).



**Şekil 3.1.** eGRASP ve iCre ifadesini sağlayan plazmid şemaları.

A: pAAV-CaMKII $\alpha$ -camgöbeği pre-eGRASP, B: pAAV-CaMKII $\alpha$ -yeşil pre-eGRASP, C: pAAV-CaMKII $\alpha$ -sarı pre-eGRASP, D: pAAV-DIO-myrTagRFP-P2A-post-eGRASP ve E: pAAV-CaMKII $\alpha$ -iCre. Şekil A-D *addgene.org* sitesinden alınmıştır.

### 3.3. İntrakraniyel Enjeksiyonlar

C57BL/6 fareler, 60-80 mg/kg ketamin ve 6-8 mg/kg ksilazin ile anestezi indüksiyonu ve %1-3 izofluran veya sevofluran ile anestezi idamesi sağlandıktan sonra stereotaksik düzleme yerleştirildi. Skalp transvers kesiyle açıldıktan sonra bregma ve lambda noktalarının aynı doğrultuda olması sağlandı. Enjeksiyon bölgeleri işaretlendi ve kraniyum bu bölgelerde drillenerek açıldı. Drilleme esnasında aşırı ısınmanın önlenmesi için sıklıkla soğuk salin uygulaması yapıldı. PE tubinge bağlı, 28 G kalınlığındaki iğneler enjeksiyon bölgesine 5-10 dakikada indirildi. İğneler enjeksiyon bölgesinde 1-2 dk bekletildikten sonra 0,1 µL/dk hızla virüs enjeksiyonu yapıldı. Enjeksiyon sonrasında virüslerin bölgeye yayılmasının sağlanması için 10 dk beklendi. Ardından kılcal etkiyle sıvının yüzeye geri tepmesinin önlenmesi için enjeksiyon iğnesi 5-10 dk içinde yavaşça geri çekildi. Kraniyotomiler kemik mumu ile kapatıldıktan sonra skalp dikildi. Yara bölgesine ve göze teramisin uygulaması, ağrı kontrolü için subkutan flunixin uygulaması yapıldıktan sonra battaniye üstünde uyanması beklendi. Uyandıktan sonra fare kafesine geri yerleştirildi. Enjeksiyon süresince farenin sıcaklığı rektal prob ile takip edildi. Keratit oluşumunu önlemek için kornea sıklıkla fosfat tamponlu salin veya Refresh Liquigel (Allergan) solüsyonlarıyla yıkandı.

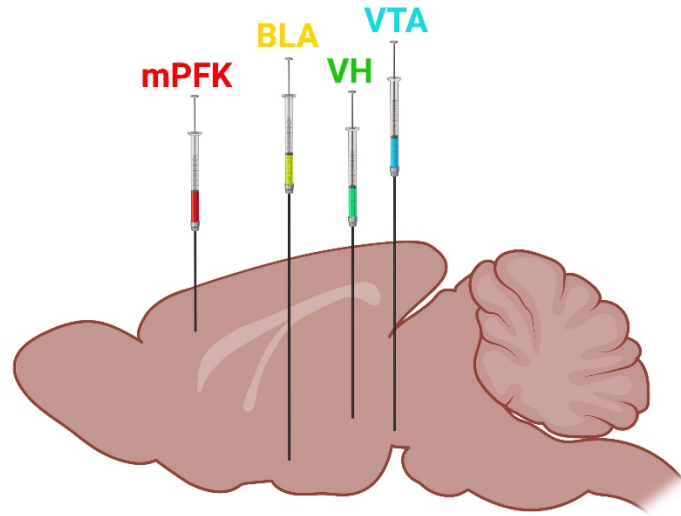
Bu tez çalışmasında, AAV2/1 – CaMKII $\alpha$  - camgöbeği pre-eGRASP, AAV2/1 – CaMKII $\alpha$  - yeşil pre-eGRASP, AAV2/1 – CaMKII $\alpha$  - sarı pre-eGRASP, AAV2/1 – DIO - post eGRASP - P2A - TagRFP-T ve AAV2/1 – CaMKII $\alpha$  - iCre içeren virüsler kullanıldı. Enjekte edilen virüsler, enjeksiyon bölgeleri ve koordinatları, hacimleri ve viral genom derişimleri aşağıda belirtilmiştir (Tablo 3.1.; Tablo 3.2.; Şekil 3.2.; Şekil 3.3.). Virüs enjeksiyonlarından sonra farelerin genel durumu kütle takibi ile izlenmiştir.

**Tablo 3.1.** İntrakraniyel enjeksiyon koordinatları (bregmaya göre).

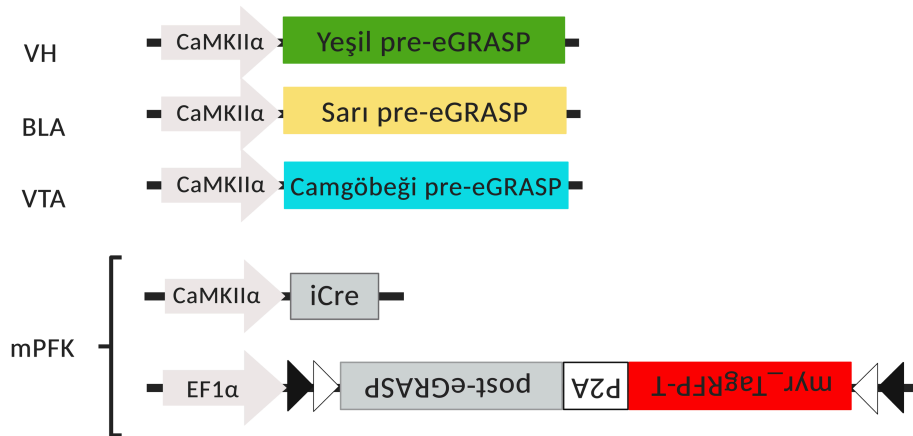
Beyin Bölgesi	Eksenler		
	AP (mm)	ML (mm)	DV (mm)
BLA	-1,20	-3,42	-5,40
VH	-3,05	-3,35	-4,20
VTA	-2,87	-0,35	-5,10
mPFK	+1,30	-0,30	-2,30

**Tablo 3.2.** Çalışmada kullanılan virüsler, derişimler, enjeksiyon bölgeleri, virüslerin protein ifadesini sağladığı nöron alt tipleri.

Virüs	Derişim (vg/ $\mu$ L)	Hacim ( $\mu$ L)	Bölge	Hücre Alt Tipi
AAV2/1 - CaMKII $\alpha$ - Sarı pre-eGRASP	$6,67 \times 10^9$	1	BLA	Piramidal Nöron
AAV2/1 - CaMKII $\alpha$ - Yeşil pre-eGRASP	$6,65 \times 10^9$	1	VH	Piramidal Nöron
AAV2/1 - CaMKII $\alpha$ - Camgöbeği pre-eGRASP	$6,8 \times 10^9$	1	VTA	Piramidal Nöron
AAV2/1 - DIO - TagRFP-T - p2A - post-eGRASP	$6,79 \times 10^8$	0,25	mPFK	Cre Bağımlı
AAV2/1 - CaMKII $\alpha$ - iCre	$4 \times 10^8$	0,25	mPFK	Piramidal Nöron



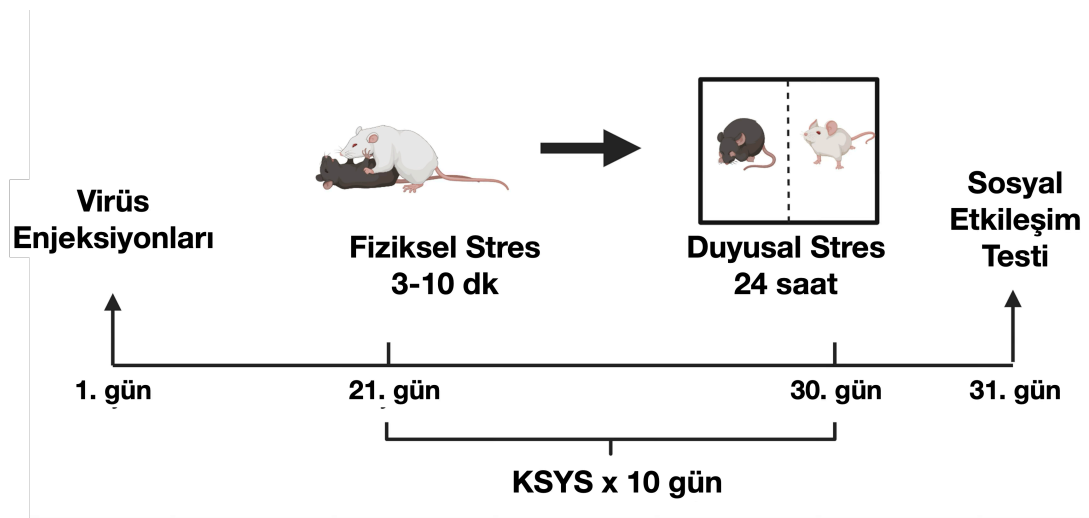
**Şekil 3.2.** Çalışmada kullanılan virüsler ve enjeksiyon bölgeleri.



**Şekil 3.3.** Çalışmada kullanılan AAV yapıları.

### 3.4. Deney Takvimi

Virüs enjeksiyonundan 3 hafta sonra stres grubu farelerine 10 gün KSYS uygulandı. 11. gün sosyal etkileşim testiyle (SET) strese duyarlılık fenotipi (strese duyarlı veya dirençli) ayrıştırıldı. Stres naif fareler virüs enjeksiyonundan sonra iyileşme sürecinde 3-7 gün kendi kafesinde tek başına, sonrasında bir engelle ayrılacak şekilde (duyusal etkileşimin mümkün ancak fiziksel etkileşimin mümkün olmadığı) kafes arkadaşıyla beraber tutuldu (118). Bu farelere 4 hafta sonra SET uygulandı. SET uygulamasından sonra fareler sakrifiye edildi (Şekil 3.4.).



Şekil 3.4. Kronik sosyal yenilgi stres deney takvimi.

### 3.5. Kronik Sosyal Yenilgi Stresi (KSYS)

Golden ve arkadaşlarının (118) kronik sosyal yenilgi stresi (KSYS) protokolü laboratuvarımıza uyarlanarak uygulandı.

#### 3.5.1. KSYS İçin Saldırgan Fare Seçimi

Deney grubu dışında 3 tane C57BL/6 cinsi erkek fare saldırgan CD-1 farelerin tespiti için kullanıldı. Saldırganlığının test edilmesinden önce CD-1 farelerin, kafesleri pleksiglas engelle ikiye ayrılması suretiyle 3 gün boyunca kendi kafeslerinin bir tarafına alıştırılması sağlandı. Sonraki 3 gün CD-1 fareler, kendi kafesi içinde her gün farklı bir C57BL/6 fare ile 3 dakikalık fiziksel etkileşime maruz bırakıldı. Bu süreçte:

- 3 günün peş peşe olacak şekilde en az 2 gününde C57BL/6 fareye saldıran fareler
- Saldırımı etkileşimin ilk bir dakikasında en az bir kez olmak üzere en az iki kez saldıran

CD-1 fareler, saldırgan olarak gruplandı ve KSYS uygulamasına dahil edildi.

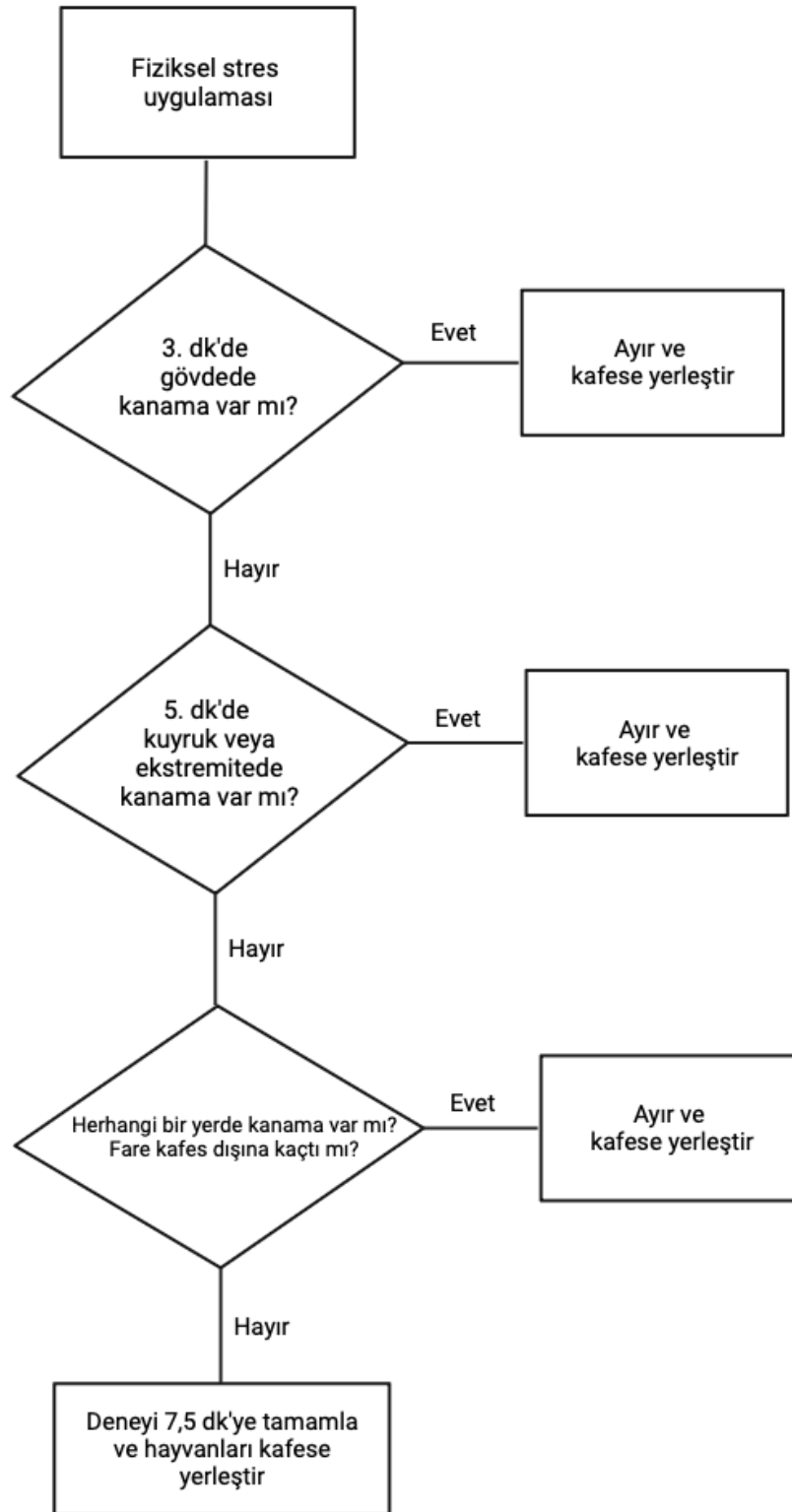
Fiziksel etkileşim sırasında görülen kovalama, sırt, anal bölge ve kuyruk bölgesini ısırma, sert şekilde tuş etme, kulak yalama ve üstüne çıkma (*mounting*) davranışları saldırganlık olarak sınıflandırıldı (119).

#### 3.5.2. KSYS Stres Uygulaması

Kronik stres grubundaki C57BL/6 cinsi fareler 10 gün boyunca 10 farklı CD-1 cinsi erkek fareyle her gün farklı bir fareyle olacak şekilde CD-1 farenin kafesinde fiziksel ve duyuşsal teması izin verecek şekilde etkileştirildi. Fiziksel etkileşim günde 3-10 dk uygulandı. Bu süreçte CD-1 farenin, yaşam alanını istila ettiğini düşündüğü C57BL/6 fareye fiziksel hasar vermek veya tehditkar davranışlar suretiyle strese soktuğu gözlemlendi. Deney sürecinde uzamış fiziksel etkileşimin C57BL/6 farenin sağ kalımında olumsuz etkiye sebep olduğu gözlemlendiği için kanama bölgesi bağımlı fiziksel stres algoritması oluşturuldu (Şekil 3.5.). Bu algoritmaya göre fiziksel etkileşim 7,5 dk ile sınırlandırıldı. Fiziksel etkileşim sırasında kuyrukta veya ekstremitelerde kanama gözlemlendiyse fiziksel etkileşim süresi 5 dk'ye; bunun dışındaki vücut bölgelerinde



kanama gözlemediyse fiziksel etkileşim süresi 3 dk'ye düşürüldü. Kanama süresi sınırından normal sonlanma süresinin sonuna kadar kanama görülmesi durumunda fiziksel etkileşim derhal sonlandırıldı. Eğer 7,5 dk içinde CD-1 fare yeterli saldırganlık davranışı göstermediyse deney süresi 10 dk'ya uzatıldı ve süre bitiminde fiziksel etkileşim sonlandırıldı. Fiziksel etkileşim sonrasında C57BL/6 fareler CD-1 farenin bulunduğu kafesin pleksiglasla ayrılan diğer bölümüne alınıp 24 saat boyunca CD-1 fareyle duyuşsal (görme, koku, ses aktarımına imkan veren) etkileşmesi sağlandı.



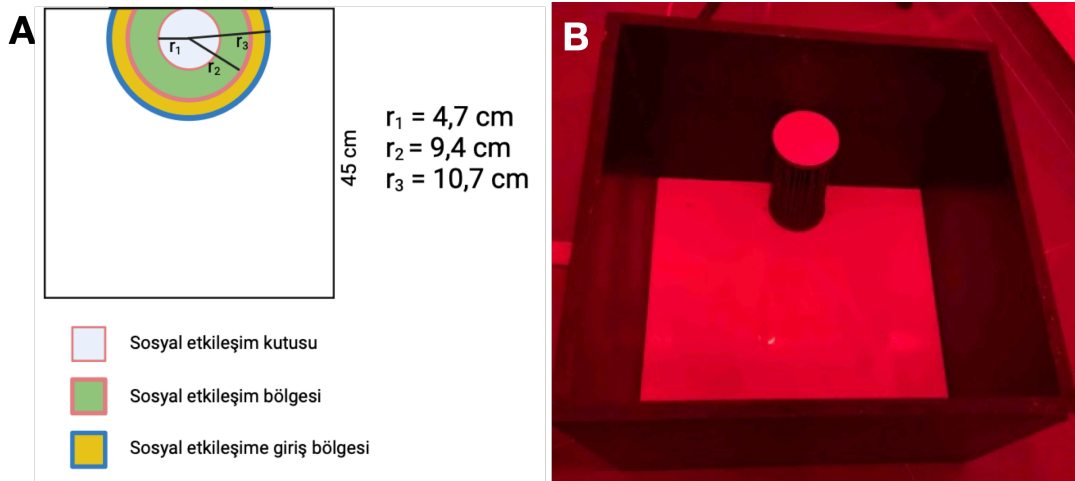
Şekil 3.5. KSYS'de kanama bölgesi bağımlı fiziksel stres uygulama algoritması.

Normal fizyolojik koşullarda kortikosteroidler memelilerde sirkadyen şekilde üretilir (120). Ancak akut stres sırasında kortikosteroid üretimi artış gösterir (121). Kortikosteroid üretiminin farede zirve yaptığı dönemde (karanlık döngü başlangıcı) verilen eksternal kortikosteroidin gece yapılan motor eğitim sonrasında motor korteks piramidal nöronlarındaki dendritik çıkıntılarının hacmini ve miktarını artırdığı tespit edilmiştir (122). Bu iki olguya dayanarak sinaptik bağlantıların güçlü şekilde eldesinin sağlanması için KSYS uygulamaları, farelerin karanlık döngüsünün başlangıcı civarında olan 18.00 – 22.00 saatleri arasında uygulandı.

### 3.6. Sosyal Etkileşim Testi (SET)

#### 3.6.1. SET Test Düzenegi

Golden ve arkadaşlarının tanımladığı sosyal etkileşim testi (SET) protokolü laboratuvarımıza uyarlanarak yapıldı (118). SET için 45 x 45 x 40 cm boyutlarında üstü açık bir kutu kullanıldı. Sosyal etkileşim kutusu için 9,4 cm çaplı, 15 cm uzunluğunda üst ve alt yüzü kapalı ancak yan yüzü uzunlamasına açık bir silindir kullanıldı. Bu silindir SET alanının bir kenarına yerleştirildi ve SET süresince aynı kenarda kalması sağlandı. SET kırmızı loş ışık altında gerçekleştirildi (Şekil 3.6.).

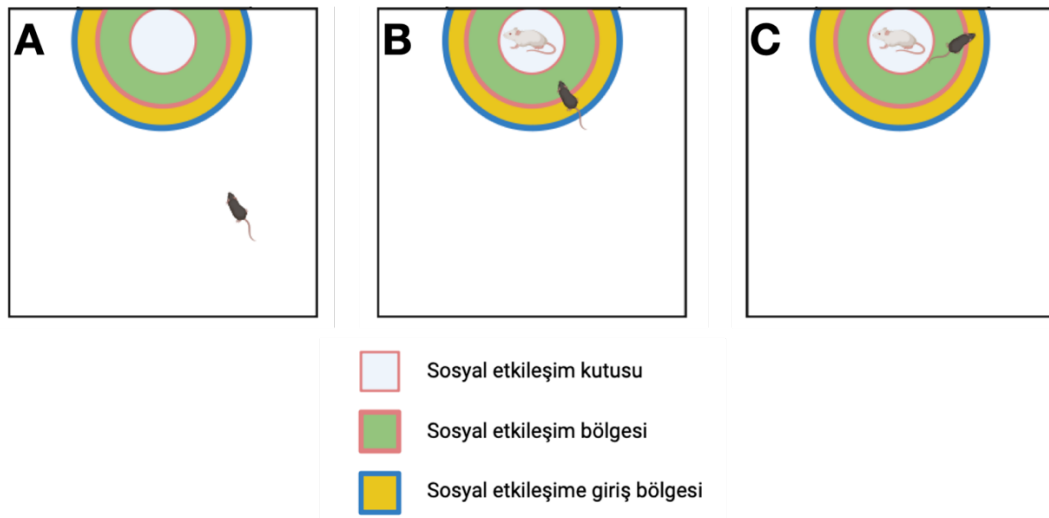


#### Şekil 3.6. Sosyal etkileşim test düzenegi.

A: Sosyal etkileşim test düzenek şeması ve B: sosyal etkileşim test düzeneginin örnek bir fotoğrafı gösterilmiştir.

SET öncesinde C57BL/6 fareler yeni kafeslerinde bir gün süreyle tek başına bekletildi ve teste başlamadan önce SET kutusunun içinde bir saat gezmesi sağlanarak ortam alıştırmaları yapıldı. Alıştırma sonrasında önce sosyal etkileşim odasında

herhangi bir fare yokken 5 dakikalık serbest hareket video kaydı tepe kamerası kullanılarak alındı. Sonra kendi kafesine 2-3 dk süreyle alındı. Bu süre zarfında CD-1 fare sosyal etkileşim odasına yerleştirildi ve izopropil alkol veya etil alkolle ortam temizliği yapıldı. Bu şekilde, sosyal etkileşim odasında CD-1 fare mevcudiyetinde 5 dk'lık serbest hareket kaydı alındı. Test sırasında C57BL/6 farenin sosyal etkileşim alanında bulunan ve bu alanda CD-1 fareye yönelen ve etkileşen fareler sosyal etkileşimde sayıldı (Şekil 3.7.). SET video kayıtları Ethovision XT 8 yazılımıyla alındı.



### Şekil 3.7. Sosyal etkileşim test uygulaması.

**A.** Sosyal etkileşim kutusunda herhangi bir CD-1 fare yokken C57BL/6 fare açık alanda serbestçe gezebilmiştir. **B-C:** Sosyal etkileşim kutusuna CD-1 fare yerleştirildiğinde sosyal etkileşim C57BL/6 farenin CD-1 fareye yönelmesine göre sınıflandırılmıştır. **B.** C57BL/6 fare CD-1 fareyle sosyal etkileşimi sağlamış; **C.** C57BL/6 fare sosyal etkileşim bölgesinde olmasına rağmen CD-1 fareyle sosyal etkileşimde değil.

#### 3.6.2. SET'in Değerlendirilmesi

Farelerin strese duyarlılık fenotipi sosyal etkileşim bölgesinde geçirilen süreye göre yapılmıştır. Sosyal etkileşim bölgesi, sosyal etkileşim odasıyla aynı merkezli ancak 2,5 katı çaplı bir daire olarak tanımlandı (Şekil 3.6.; Şekil 3.7.). Eğer C57BL/6 farenin yönelimi CD-1 fareye doğruysa ve bu alanın içindeyse C57BL/6 fare CD-1 fareyle sosyal etkileştiği düşünüldü (Şekil 3.7.). Sosyal etkileşim oranı aşağıdaki şekilde tanımlanmıştır:

$$\text{Sosyal Etkileşim Oranı} = \frac{t_{dolu}}{t_{boş}} \quad (3.1.)$$

$t_{dolu}$ , sosyal etkileşim kutusunda CD-1 fare mevcudiyetinde C57BL/6 farenin CD-1 fareyle etkileştiği süreyi;  $t_{boş}$ , sosyal etkileşim kutusunda CD-1 fare yokken C57BL/6 farenin sosyal etkileşim bölgesinde geçirdiği süreyi ifade etmektedir. Sosyal etkileşim oranı 1'den büyükse C57BL/6 farenin strese dirençli, 1'den küçükse strese duyarlı fenotipte olduğu kabul edilmiştir.

### 3.7. Sakrifikasyon ve Doku Hazırlığı

Tüm farelerde %10'luk kloral hidrat ile anestezi indüksiyonu sağlandıktan sonra %4'lük paraformaldehit (PFA) çözeltisiyle intrakardiyak perfüzyon yapıldı. Çıkarılan beyin +4 °C sıcaklıkta, %4'lük PFA içinde 1-2 gece bekletildi. Ardından 3 gün boyunca fosfat tamponlu salinde (PBS) yıkandı. Sonra %30'lük sükröz çözeltisinde 2-5 gün bekletildi. Altmış µm'lik donmuş yüzen (free-float) kesitler alındı. Kesitler 1-3 gün PBS içinde bekletildikten sonra lama yayıldı ve gliserol/PBS:50/50 solüsyonuyla kapatıldı.

### 3.8. Görüntüleme

Görüntüler Hacettepe Üniversitesi Nörolojik Bilimler ve Psikiyatri Enstitüsü Beyin Araştırma Laboratuvarında bulunan Leica SP8 konfokal mikroskop ve Ankara Üniversitesi Tıp Fakültesi Histoloji ve Embriyoloji Anabilim Dalında bulunan Zeiss LSM810 konfokal lazer taramalı mikroskoplarla alındı. Sinaptik sinyallerin doğru şekilde tespit edilmesi için yüksek çözünürlükte görüntüleme sağlayan yüksek numerik apertürlü (NA) objektifler kullanıldı (63x yağlı, NA: 1,4 ve 40x yağlı, NA: 1,3).

Görüntüleme esnasında önce sinaptik sinyallerin ifadesinin sağlandığı doğrulandı. Sonrasında multispektral görüntünün hızlı şekilde kazanımı için çerçeve sıralı (frame sequential) görüntüleme düzenleme yapıldı. Son olarak yüzeyden derine tam kolon görüntüsünün kazanılması için kiremit tarama (*tile-scan*) sistemi düzenlendi.

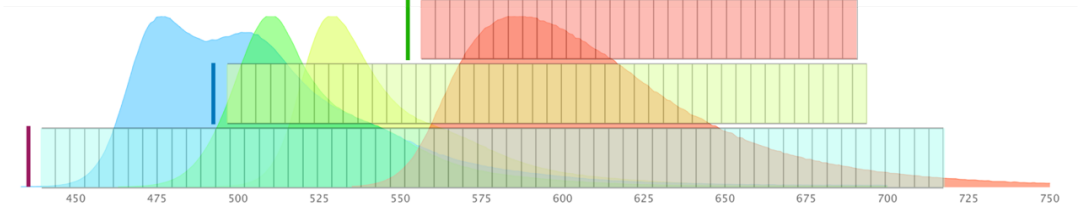
#### 3.8.1. Sinaptik Sinyallerin Doğrulanması

Yeşil ve sarı eGRASP sinyallerinin emisyon intensite eğrileri birbirlerine yakın olduğu için iki sinyalin birbirinden ayrıştırılması gerekiyordu. Bunun yanı sıra hem

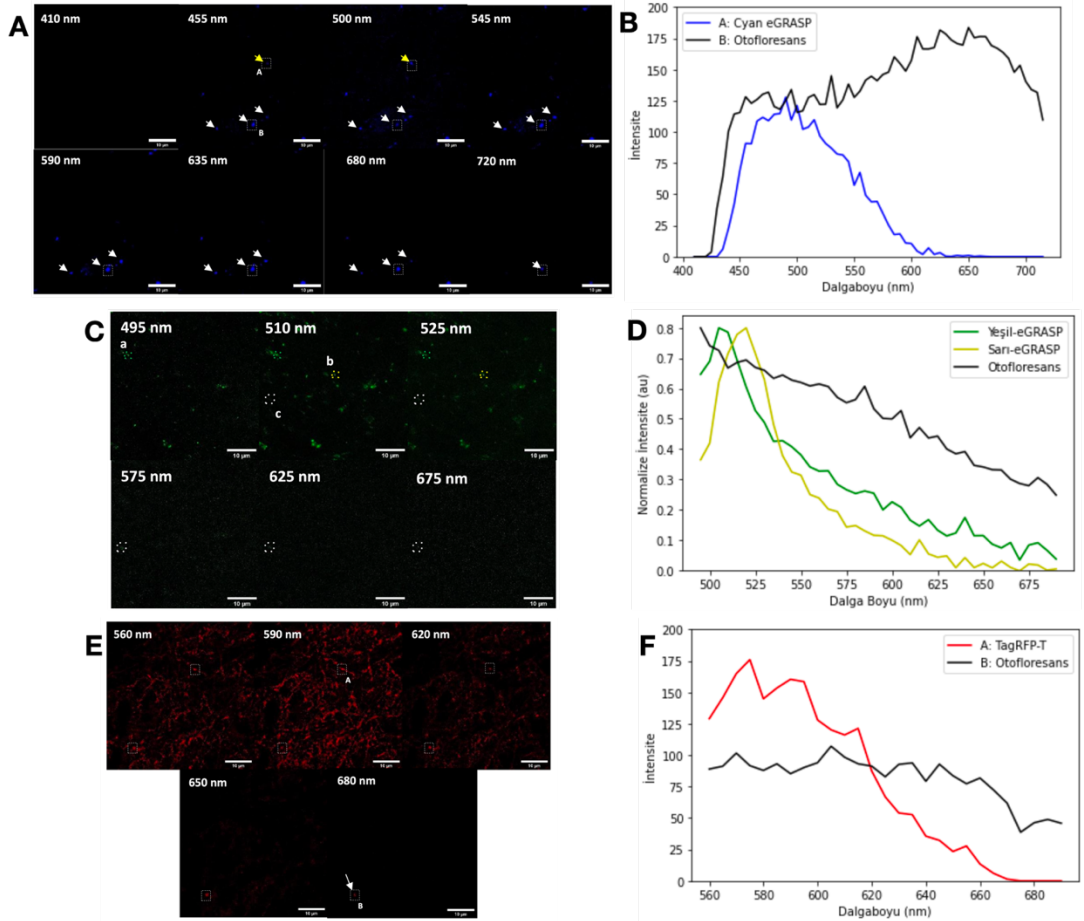
literatürdeki eGRASP çalışmalarında hem de çalışma sırasındaki görüntülerin incelenmesi sırasında ağırlıklı olarak çekirdekten kaynaklı otofloresans sinyallerin sinaptik sinyal benzer boyutlara sahip olabileceği gözlemlendi (123). Otofloresans sinyaller bütün sinyal pencerelerinde sinyal oluşturabildiği için bu sinyallerin sinaptik sinyallerden ayrıştırılması gerekti.

İncelenen sinyalin sinaptik sinyal içerip içermediğinin tespit edilmesi için dalga boyu tarama (*lambda scanning*) yöntemi kullanıldı. Dalga boyu taraması camgöbeği eGRASP için 405 nm; yeşil-sarı eGRASP için 488 nm ve TagRFP-T proteini için 552 nm eksitasyonu sonrasında, camgöbeği eGRASP için 410-720 nm; sarı-yeşil eGRASP için 495-695 nm ve TagRFP-T proteini için 560-690 nm arasındaki sinyaller 5 nm'lik aralıkla band-pass filtreden geçirilerek toplandı. Sonrasında alınan sinyallerin kendi içlerinde emisyon eğrileri çıkarıldı, ardından birbirlerine göre normalize edildikten sonra karelerinin farkını minimize edecek lineer ayrıştırma işleminden geçirildi (124, 125) (Şekil 3.8.).

Lineer ayrıştırma sonucunda camgöbeği, yeşil ve sarı eGRASP sinyalleri ve TagRFP-T protein sinyalleri hem birbirlerinden hem de otofloresanstan başarılı şekilde ayrıştırıldı. Ayrıştırma sonucunda Choi ve Kaang'ın tarifine uyumlu şekilde (126) farklı eGRASP proteinlerinin emisyon maksimum dalga boylarıyla uyumlu şekilde camgöbeği eGRASP'ın ~470 nm; yeşil eGRASP'ın ~510 nm, sarı eGRASP ~530 nm ve TagRFP-T'nin ~580-585 nm'de maksimum parlaklıktaki sinyali oluşturduğu gözlemlendi (Şekil 3.9.).



Şekil 3.8. Dalga boyu taramalı görüntüleme şeması.



Şekil 3.9. Örnek dalga boyu tarama görüntü sonuçları.

A-B: Camgöbeği eGRASP'ın 405 nm eksitasyon altındaki dalga boyu tarama görüntüleri ve emisyon dalga boyu eğrisi. C-D: Yeşil ve sarı eGRASP'ın 488 nm eksitasyon altındaki dalga boyu tarama görüntüleri ve emisyon dalga boyu eğrileri. E-F: TagRFP-T floresan proteininin 552 nm eksitasyon altındaki dalga boyu tarama görüntüleri ve emisyon eğrisi. Tüm eğriler aynı bölge bulunan otofloresans sinyallerle beraber çizilmiştir. Ölçek: 10 µm.

### 3.8.2. Sinaptik Sinyallerin k-Ortalamlar Yöntemiyle Ayırıştırılması için Çerçeve Sıralı Görüntü Alımı

Her ne kadar multispektral görüntülemelerde lineer ayırıştırma birbirlerine yakın emisyon eğrileri olan sinyallerin ayırıştırılması için kullanılsa da z-stack alımı sırasında 5 nm pencereci veri kazanımı sırasında yeşil ve sarı eGRASP sinyalleri normalden uzun süre uyarılır. Bu da sinaptik sinyallerin z-stack'lerin derininde solmasına sebep olmaktadır. Bu durumu aşmak için görüntü alım sıklığı azaltılabilir ve elde edilen bilgi makine öğrenmesi temelli spektral ayırıştırma algoritması geçirilebilir (127).

Literatürde lineer ayırıştırma kaynaklı bu kısıtlılığın çözülmesi için makine öğrenmesi tabanlı birçok yöntem tarif edilmiştir (127-130). Ancak birçok yöntem ya çok büyük setleri için hazırlanmış, ya bilgisayar üzerinden işlenmesi karışık ve uygulanamaz hale getirilmiş ya da spesifik mikroskopi tekniklerine göre hazırlanmıştır. Bu kapsamda çalışmada hem kullanımı rahat hem de kendi veri alımımıza kolaylıkla uygulayabilen bir yöntem olan *Learning Unsupervised Means of Spectra* (LUMoS)'nın kullanılması tercih edildi (127).

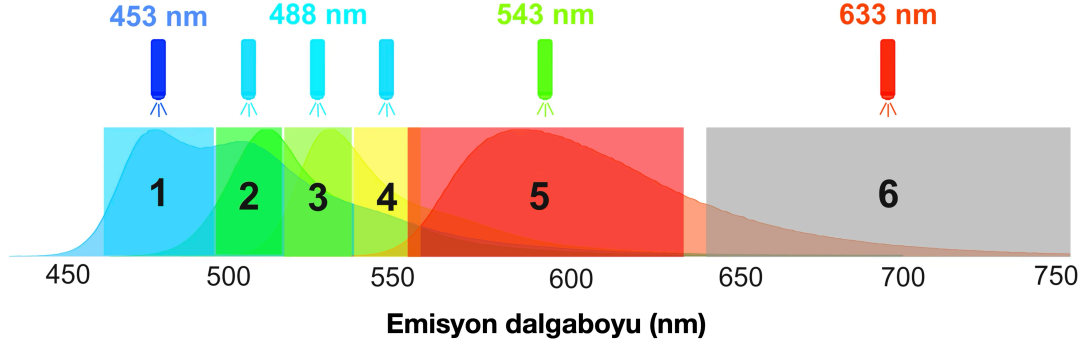
LUMoS algoritması, önce referans bir görüntüde floresans pencerelerdeki göreceli emisyon farklılıklarını kullanarak bir pikselin hangi fluorofora ait olduğunu k-ortalamlar yöntemiyle tayin eder. Ardından tayin edilen fluoroforların o pikseldeki kesitsel spektral bilgisine göre 3 boyutlu görüntüdeki diğer sinyallerin hangi florofora ait olduğunu eğitimsiz ve kör (*unsupervised blind unmixing*) şekilde tespit eder.

LUMoS spektral ayırıştırmanın başarıyla yapılabilmesi için, en az tespit edilmesi gereken fluorofor kadar spektral bilgi gereklidir. Bu sebeple çalışmada otofloresans dahil 5 tane fluorofor incelendiğinden dolayı en az 5 kesitli spektral görüntüleme yapılması gereklidir. Bunun sağlanması için konfokal mikroskopun çerçeve sıralı (*frame sequential*) görüntüleme özelliği kullanıldı.

Çerçeve sıralı görüntüleme esnasında hem tarafımızca tespit edilen hem de Choi ve Kaang'ın (126) her bir eGRASP sinyali tarif ettikleri optimal emisyon dalga boyları pencereler içinde tutularak planlandı. Çerçeve sıralı görüntüleme camgöbeği eGRASP için 453 nm eksitasyon sırasında sırasıyla 460-490 nm'lik emisyon pencerelerinden; yeşil-sarı eGRASP için 488 nm eksitasyon sırasında 495-515, 515-535 ve

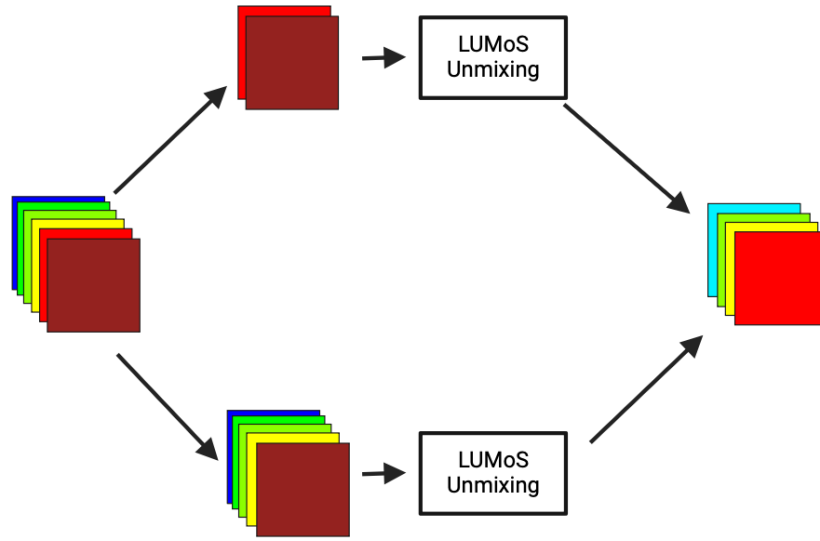


535-555 nm'lik pencerelerden, dendritlerde ifade edilen TagRFP-T proteini için 552 nm eksitasyon sırasında 560-630 nm'lik emisyon penceresinden ve otofloresans proteinlerin eldesi için 632 nm eksitasyon sırasında 650-800 nm'lik emisyon penceresinden görüntü toplandı (Şekil 3.10.).

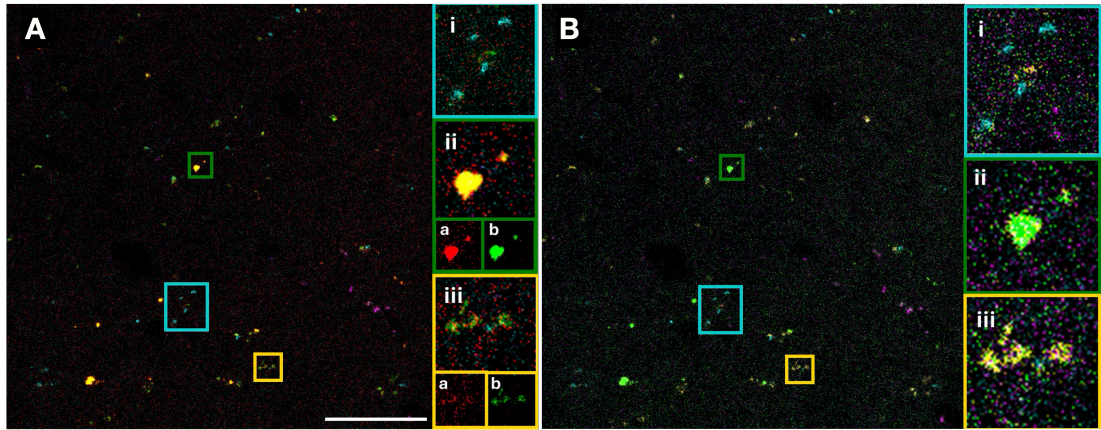


**Şekil 3.10.** Çalışmada uygulanan çerçeve sıralı görüntüleme yöntemi.

Alınan görüntü işlenmek üzere dendritik ve sinaptik olmak üzere iki gruba ayrıldı. Dendritik grubundan dendrit ve otofloresans görüntüleme için kullanılan görüntüler; sinaptik görüntü grubunda sinapslar ve otofloresans görüntüleme için kullanılan görüntüler olacak şekilde düzenlendi. Sonra LUMoS kullanılarak dendritik görüntü grubu 2 florofora (TagRFP-T ve otofloresans); sinaptik görüntü grubu 4 otoflofora ayrıştırıldı (camgöbeği, yeşil, sarı ve otofloresans) (Şekil 3.11.). Bu işlemler sonucunda hem dendritik hem sinaptik sinyaller başarılı şekilde ayrıştırıldı (Şekil 3.12.).



**Şekil 3.11.** Çerçeve sıralı görüntünün k-ortalamlar yöntemiyle ayrıştırılması için kullanılan algoritma.



**Şekil 3.12.** LUMoS spektral ayrıştırma öncesi ve sonrası örnek sinaptik sinyaller.

**A:** Şekil 3.10'da gösterilen çerçeve sıralı görüntü alımı sonrası sinapsların görüntüsü gösterilmiştir. **i:** 453 nm lazer eksitasyonu sonrasında görülen camgöbeği eGRASP görüntüsü, **ii:** 488 nm lazer eksitasyonu sonrasında görülen yeşil eGRASP görüntüsü. 488 nm lazer eksitasyonu sonrasında 495-515 nm emisyon penceresinde **(a)** yeşil eGRASP sarı eGRASP'a göre daha fazla foton emisyonu sağlar ve daha parlak görünür. Yeşil eGRASP 515-535 nm emisyon penceresinde **(b)** sarı eGRASP'a benzer düzeyde sinyal oluşturabildiği için bu emisyon penceresinde de parlak görünür. **iii:** 488 nm lazer eksitasyonu sonrasında görülen sarı eGRASP görüntüsü. 488 nm lazer eksitasyonu sonrasında sarı eGRASP 495-515 nm emisyon penceresinde **(a)** sönük sinyal oluşturur. Ancak 515-535 nm emisyon penceresinde **(b)** güçlü bir sinyal oluşturabilir. **B:** LUMoS spektral ayrıştırma sonrası sinaptik görüntüler. **i:** camgöbeği, **ii:** yeşil ve **iii:** sarı eGRASP. Ölçek: 25  $\mu\text{m}$ .

### 3.8.3. Kesit Yüzeyinden Derine Doğru Çerçeve Sıralı Görüntü Alımı

Yapılan çalışmalarda sinapsın alansal büyüklüğünün  $0,055 - 0,1 \mu\text{m}^2$  arasında değiştiği gösterilmiştir (131). Buna göre bir sinapsın ışık mikroskopisinde başarılı şekilde tespit edilmesi için uygun çözünürlükte görüntü alınması gereklidir. Objektiflerin piksel rezolüsyon limitleri Rayleigh rezolüsyon kriterleriyle belirlenir. Buna göre bir objektifin xy düzleminde ve z-eksenindeki çözünürlüğü sinyalin dalga boyuna, numerik apertüre ve immersiyon sıvısının kırıcılık indisine bağlıdır (132).

xy düzleminde Rayleigh rezolüsyon limiti aşağıdaki formülden hesaplanır:

$$xy_{confocal} = \frac{1,22\lambda}{2NA} \quad (3.2)$$

z-ekseninde Rayleigh rezolüsyon limiti aşağıdaki formülden hesaplanır:

$$z_{confocal} = \frac{2\lambda\eta}{NA^2} \quad (3.3)$$

$\lambda$ , dalga boyunu;  $\eta$ , immersiyon sıvısının kırıcılık indisini ve NA, numerik apertürü ifade etmektedir.

Yukarıdaki formüller elimizdeki objektiflere ve proteinlerin tahmini emisyon maksimum dalga boylarına uygulandığı zaman Tablo 3.3. elde edildi.

**Tablo 3.3.** eGRASP proteinleri ve TagRFP-T floresan proteininin 40x ve 63x yağlı objektifler için Rayleigh çözünürlük sınırları

Proteinler	Emisyon Maksimum Dalga Boyu (nm)	Rayleigh Çözünürlük Sınırı (nm)			
		40x, Yağlı NA: 1,3		63x, Yağlı, NA: 1,4	
		xy	z	xy	z
Camgöbeği eGRASP	480	225	862	209	743
Yeşil eGRASP	509	239	914	222	788
Sarı eGRASP	527	247	946	230	816
TagRFP-T	585	274	1051	255	906

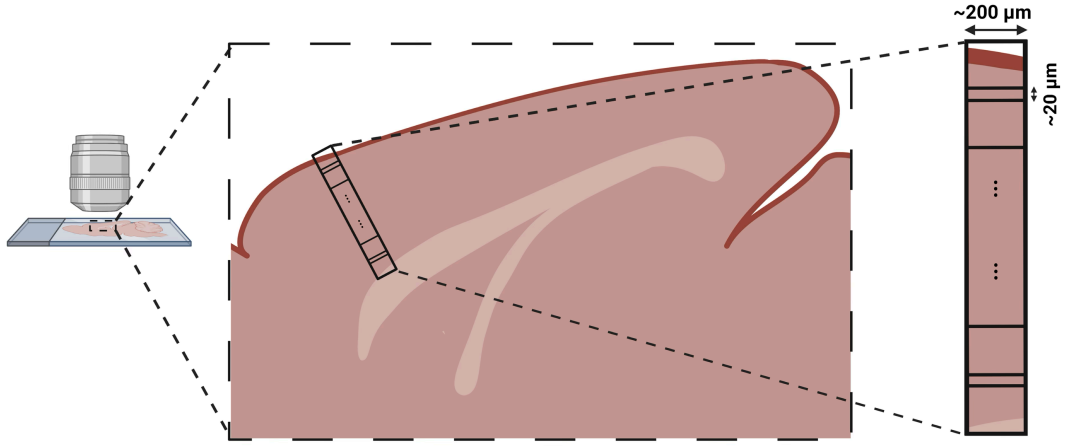
Yukarıdaki tablodaki çözünürlük değerleri kullanılarak mikroskop büyütmesi düzenlendi. Görüntü alımı sırasında piksel mesafesi ve derine görüntüleme esnasındaki z basamak uzunluğu hesabında Nyquist teorisi de göz önünde tutuldu. Görüntüleme esnasında her bir çerçeve sırasında piksel mesafeleri eşit şekilde ayarlanması ve bu ayarlama esnasında veri kazanım sıklığının sinyal kaybını minimize edecek şekilde düzenlenmesi gereklidir. Bundan dolayı piksel mesafesini camgöbeği eGRASP sinyali xy ve z piksel mesafesini yeterli düzeyde toplayacak; ancak otofloresans pencerede fazla veri toplayacak (*oversampling*) şekilde düzenlendi. Sonuçta derine doğru görüntü alımı yapılırken xy piksel mesafesi ~100 nm, z basamak mesafesi ~300 nm şekilde düzenlendi. Çerçeve sıralı görüntü alımı esnasındaki emisyon sinyal pencereleri optimizasyon protokolündeki emisyon pencereleriyle aynı olacak şekilde düzenlendi. Yüzeyden derine doğru geçilirken lazer uyarım şiddetinde derinlik bağımlı bir değişiklik yapılmadı.

#### 3.8.4. Kiremit Taramalı Görüntüleme

Bu projede mPFK yüzeyinden korpus kallosuma uzanan kortikal kolon görüntü alınması hedeflendi. Fare korteksinin çalışmada kullanılan objektiflerin görüntüleme alanından (*field of view, GA*) daha geniş olması ve derine doğru çerçeve sıralı görüntülemenin tek tek sıralı şekilde yapılması yorucu bir teknik olmasından ötürü

korteks yüzeyinden korpus kallozuma otomatik şekilde kiremit taramalı görüntüleme yapıldı (Şekil 3.13.).

Kiremit sıralı görüntüleme işlemi korteks yüzeyinden korpus kallozuma doğru GA'nın yaklaşık %10'u çakışacak şekilde tek sıra halinde alındı. Tüm kiremit parçalarında derine doğru çerçeve sıralı görüntüleme işlemi gerçekleştirildi (Şekil 3.13.). GA bu süreçte  $\sim 205 \times 205 \mu\text{m}^2$  olacak şekilde ayarlandı.



**Şekil 3.13.** Kiremit taramalı görüntüleme şeması.

### 3.8.5. Reflektans Görüntüleme

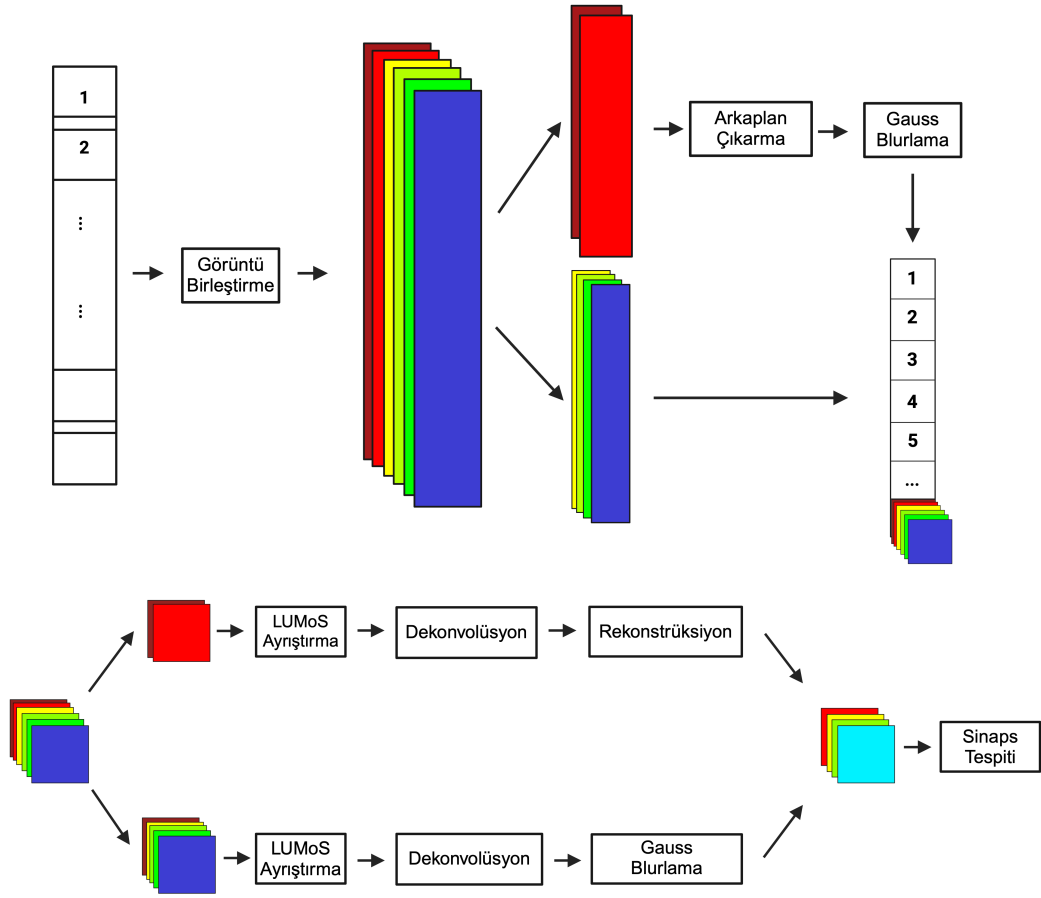
6. katman ve ventralindeki beyaz cevherin arasındaki sınırın belirlenmesi için konfokal reflektans görüntüleme uygulandı. Kısaca 488 nm emisyon sonrasında 483-493 nm arasında dar emisyon pencereci görüntü sinyalleri, RT 15/85 ışın ayırıcı (*beam splitter*) ile kazanıldı. Bu süreçte hem kiremit taramalı hem de 3 boyutlu görüntüleme yapıldı. Bu görüntüleme esnasında görüntü yüzeylerinin arasındaki z-eksen mesafe 2-3  $\mu\text{m}$  civarı şekilde; xy piksel boyutu da  $\sim 0,6 \mu\text{m}$  şekilde düzenlendi. Bu görüntülemeye ek korteks üzerindeki TagRFP-T sinyali de 552 nm uyarım sonrasında kazanıldı ve reflektans görüntüsüyle karşılaştırıldı. En son LASX v3.5.5.19976 yazılımındaki görüntü birleştirme algoritmasıyla kiremit tarama görüntüleri birleştirildi.

### 3.9. Görüntü İşleme

Kolon görüntüleri önce birleştirildi (*stitching*). Görüntüler elimizdeki bilgisayarın merkezi hafıza (RAM) kapasitesine göre yaklaşık eşit boyutlu parçalara ayrıldıktan sonra sinaptik ve dendritik gruplara ayrıldı. Sinaptik ve dendritik gruplar “3.8.2. Sinaptik Sinyallerin k-Ortalamalar Yöntemiyle Ayrıştırılması için Çerçeve Sıralı Görüntü Alımı” başlığındaki grup ayırımına göre gruplandı.

Sinaptik gruptaki sinyaller LUMoS ile ayrıştırıldıktan, piksel boyutları izotropik yapıldıktan sonra Ricardson-Lucy (RL) dekonvolüsyon işlemine tabii tutuldu. Dekonvolve sinaps görüntüleri Gauss filtreleme ile düzeltildi.

Dendritik grup sinyalleri önce arka plan çıkarma ve Gauss filtrelemeden geçirildi. Sonra dendritik sinyaller otofloresans sinyallerden LUMoS ile ayrıştırıldı. Piksel boyutları izotropik yapıldıktan sonra RL dekonvolüsyon işlemi uygulandı (Şekil 3.14.).



Şekil 3.14. Çalışmada uygulanan görüntü işleme algoritma sıralaması.

### 3.9.1. Görüntü Birleştirme

Kiremit tarama görüntüleri birleştirilirken elde edilen görüntülerin birleştirilmesinde önce Zeiss konfokal mikroskopun kendi yazılımı kullanılmıştır. Birleştirilmiş görüntünün çakışma alanlarında sinyal duplikasyonu gözlenmişse BigStitcher Fiji plug-ini kullanılmıştır (133).

Görüntü birleştirme esnasında Zeiss konfokal mikroskopun görüntü birleştirme algoritmasındaki korelasyon eşiği genellikle 0'a veya 0.01'e çekilerek görüntü birleştirmesi yapılmıştır.

BigStitcher algoritması kullanıldığında, görüntüler önce BigDataViewer plug-inine yüklendi. Burada çakışma alanları ve bu alanlardaki görsel ipuçları kontrol edilip elle kaba birleştirme işlemi yapıldı. Sonra global optimizasyon ve iteratif yakın nokta bulma algoritmaları kullanılarak iki kiremit tarama görüntüsünün çakışma alanları korele edildi. Görüntülenen sinyaller sinaps içermesi ve sinapsların 3 boyutlu uzayda dağınık şekilde bulunması sebebiyle korelasyon eşiği 0'a çekildi. Birleştirilmiş görüntü, geçiş bölgelerinde veri kaybı riskinin ve birleştirme hata ihtimalinin azaltılması amacıyla sırasıyla düşük desimizasyon (1 piksel) ve gevşek birleştirme yöntemiyle birleştirildi. Birleştirilen görüntüdeki voksel boyutları korundu ve z-ekseninde herhangi bir interpolasyon uygulanmadı.

### 3.9.2. Spektral Ayırıştırma

Altı kanallı birleştirilmiş görüntü oldukça büyük olduğu için laboratuvarımızda bulunan bilgisayarın yeterliliğine (RAM kapasitesi: 32 GB) göre görüntüler yaklaşık eşit boyutlu parçalara bölündü. Sonra dendritik gruptaki görüntüler önce arka plan çıkarma (Yuvarlak top çapı: 50 piksel) işleminden sonra Gauss filtrelemeden (pencere büyüklüğü 1 piksel) geçirildi. Bu süreçte sinaptik gruptaki görüntülere herhangi bir ön hazırlık uygulanmadı. Bölünmüş çerçeve sıralı görüntüler LUMoS spektral ayırıştırmasından geçirildi (127). Spektral ayırıştırma için kullanılan referans görüntüler incelenecek 3 boyutlu görüntü içerisinden seçildi. Sinaptik grupta ayırıştırma yapılırken camgöbeği, yeşil, sarı eGRASP ve otofloresans sinyallerin aynı anda ifadesinin gözleendiği kesit; dendritik grupta ayırıştırma yapılırken dendritik ve otofloresans sinyallerin aynı anda görüldüğü kesitler spektral ayırıştırma için seçildi.

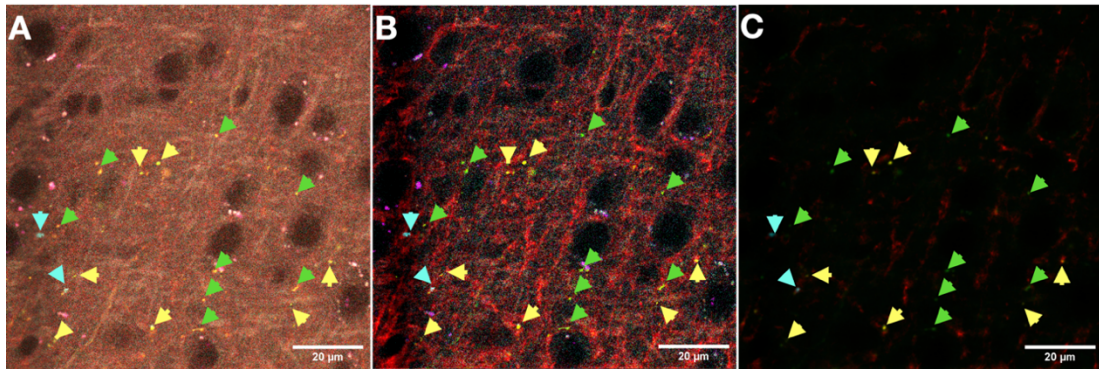


Spektral ayrıştırma işlemi en az 50 kez ve en az 100 iterasyonla yapıldı. Ortaya çıkan görüntüdeki sinyaller daha önceden gözle etiketlenmiş floresans grubuna göre etiketlendi ve kaydedildi.

### 3.9.3. Dekonvolüsyon

Hem sinaptik hem de dendritik görüntülerin vokselleri, z-ekseninde bikübik interpolasyonla izotropik hale getirildi. Sonra, grafik işlemci (*graphical processing unit, GPU*) kullanan açık erişimli Richardson – Lucy (RL) dekonvolüsyon algoritmasıyla dekonvolüsyon işlemi gerçekleştirildi (134). RL dekonvolüsyonu için kullanılacak nokta dağılım fonksiyonu (point spread function, PSF) Diffraction PSF 3D Fiji Plug-in'i kullanılarak her bir sinyalin en optimal düzeyde parlaklık veren dalga boyu için üretildi (126, 135).

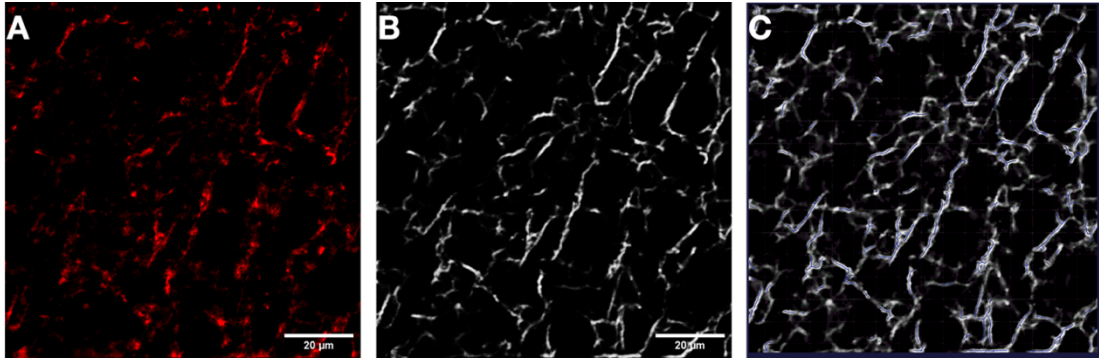
Tüm sinyaller için RL dekonvolüsyonu 3 boyutlu şekilde 20 iterasyonla yapıldı. Dendritik grup görüntülerindeki sinyaller herhangi bir ek işlem yapılmadan 32-bitten 8-bite dönüştürülüp kaydedilirken; sinaptik gruptaki görüntülerin önce 32-bit görüntüleri elde edildi. Bu görüntülere Gauss filtre (pencere büyüklüğü: 2 piksel) uygulandıktan sonra oluşan görüntü 8-bite dönüştürülüp kaydedildi (Şekil 3.15.).



**Şekil 3.15.** Spektral ayrıştırma öncesi, sonrası ve dekonvolüsyon sonrası örnek sinyaller.

### 3.9.4. Dendritik Sinyallerin Segmentasyonu ve Rekonstrüksiyonu

Dekonvolüsyon sonrasında tüm dendritik ve sinaptik görüntüler herhangi bir işlem yapılmadan dorso-ventral eksenindeki karşılığına göre birleştirildi. Dendritlerin dijital rekonstrüksiyon ağacı eldesi için Gliko ve arkadaşlarının (136) tanımladığı algoritma kullanıldı. Dendritik görüntüler, önce çeşitli mikroskopi görüntüleriyle eğitilmiş U-Net mimarisine sahip evrişimli sinir ağı (*convolutional neural network*) tabanlı algoritmayla herhangi bir hiperparametre düzenlemesi yapılmadan segmente edildi. Ardından arkaplan eşikleme, bağlı bileşen analizi (*connected component analysis*) ve iskeletleme uygulanarak *.swc* uzantılı dijital rekonstrüksiyon elde edildi. Sonrasında hatalı sinyallerin eliminasyonu için budama (*pruning*) ve görsel kontrolün sağlanması için indirgeme (*downsampling*) işlemi de uygulandı. Bu süreçte elimizdeki bilgisayarın işlemsel kaynağı yetersiz olduğu için dendritlerin çapları hesaplanmadı. Ayrıca dendritlerin soma ile açık bağlantıları net şekilde gözlemlenemediği için dendritler alt sınıflarına (apikal, bazal, oblik vs.) ayrılmadı (Şekil 3.16.).



**Şekil 3.16.** Dekonvolüsyon sonrası dendritik görüntülerin örnek segmentasyonu ve rekonstrüksiyonu.

### 3.9.5. Dendritik Segment Tanımı

Çalışmada çok sayıda nöronun dendritleri TagRFP-T ile işaretlendiği için, mikroskopi görüntüleri çalıda benzer şekilde karmaşık görünmektedir. Bu probleme ek, konfokal mikroskopun çözünürlük sınırı da 3 boyutta çapraz şekilde ilerleyen dendritlerin keskin şekilde ayırt edilmesini güçleştirmektedir. Bu sebeple, hata yapmamak için, dendritik segmentler, *.swc* rekonstrüksiyon ağacında iki dallanma noktası arasındaki dendrit parçası olarak tanımlanmıştır. Bu süreçte kiremit kenarında

gözlenen kenar artefaktları ve *.swc* ağacı içerisinde gözlenen 0  $\mu\text{m}$  uzunluklu tek nodlu segmentler de filtrelenmiştir.

### 3.10. Sinaptik Sinyallerin Yarı Otomatik Şekilde Tespit Edilmesi

Sinaps sayılarının ve dağılımlarının incelendiği çoğu çalışmada sinapslar görsel olarak incelenir ve sinaps sayıları elle tespit edilir (29, 137). Ancak elle sayım hem teknikleri giderek büyümekte olan nörobiyoloji verilerinin incelenme süresini binlerce saate çıkarabilmektedir hem de güvenilir değildir (137, 138). Kortikal kolondaki sinaps dağılımları hem veri kapasitesi hem de sayısal açıdan büyük veridir. Ancak bu verinin de kısa zamanda hızla işlenmesi gerekmektedir. Bu durumun çözülmesi için çok sayıda otomatize sinaps tespiti sağlayan otomatik algoritma üretilmiştir (139-141). Bu kapsamda, uygulaması oldukça kolay olan Synapse Quantifier algoritmasıyla sinapslar otomatik şekilde tespit edildi (140).

#### 3.10.1. Synapse Quantifier Algoritması

Synapse Quantifier sinapsların tespiti için adaptif yıkama segmentasyon (*adaptive watershed segmentation*) algoritması kullanılmaktadır. Bu işlemin esası dendritik ağaçtan belirli bir mesafeye kadar dendritik ve sinaptik sinyallerin çakışmasının tespitine dayanır (140).

Synapse Quantifier algoritması 8 bit RGB görüntüyü ve rekonstrüksiyon ağacını algoritma girdisi olarak alır. Uygulama içerisinde kullanıcı dendritlerin ve sinapsların bulunduğu görüntü katmanlarını seçer. Ardından kullandığı mikroskopun ve incelediği sinyalin özelliklerine göre dendritik ve sinaptik sinyallerin intansite eşikleri, tespit edilecek minimum hacim ve dendrit dalından itibaren incelenecek minimum uzaklık bilgilerini algoritmaya verir. Sonrasında kullanıcı sinapsları tek tek inceleyerek veya çok az bir zaman inceleyerek incelenen sinyallerin sinaps olup olmadığına veya dendrit üzerindeki konumlarına (dendritik çıkıntı üzerinde olup olmadığına) karar verir.

Synapse Quantifier algoritması, sinapsın uzaysal konumunu, voksel cinsinden hacmini, bulunduğu dendritteki en yakın *.swc* nodu ve sinapsın dendrite göre konumunu (dendritik çıkıntı üzerinde olup olmadığını) çıktısını verir.

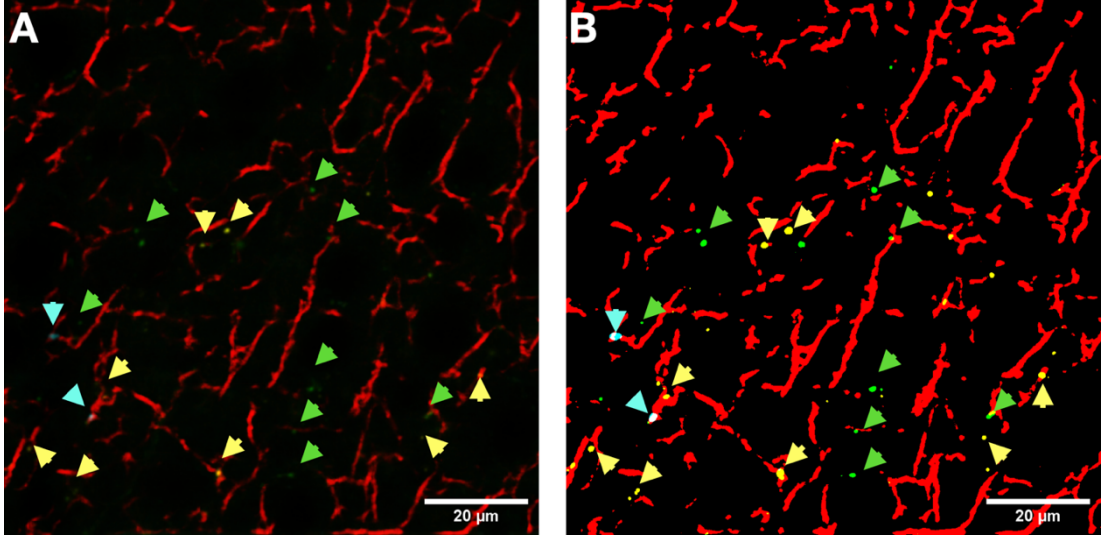
Yaptığımız gözlemler neticesinde sinaptik ve dendritik sinyallerin zaman zaman birbirleriyle çakışmayabileceğini gözlemledik. Bu durum Synapse Quantifier uygulaması sırasında sinapsın tespit edilememesi veya tespit edilen sinapsın koordinatının ve hacminin hatalı tespit edilmesi risklerini ortaya çıkarır. Bu sorunu çözmek için Synapse Quantifier algoritmasına alınacak sinyallere ön hazırlık işlemi uyguladık.

### 3.10.2. Sinaptik Sinyallerin Synapse Quantifier Algoritması İçin Hazırlanması

Segmente edilmiş dendritik sinayaller ve sinaptik sinyaller önce ayrıştıma sonucunda ortaya çıkan sinaptik sinyallerle uyumlu olacak şekilde elle eşiklemeden (*manual thresholding*) geçirildi (Şekil 3.16.). Her bir görüntünün sinyal eşiği her fare ve her bir kiremit tarama görüntü parçasının spektral ayrıştırılmış sinyaliyle karşılaştırılarak ayrı ayrı belirlendi. Eşik belirlendikten sonra her iki sinyal grubu da yüksek geçirimli filtreden (*high-pass filter*) geçirilerek eşikleme (*thresholding*) uygulandı (). Eşikleme sonrasında her bir sinyalin intansitesi 255 olarak düzenlendi.

$$I_{x,y,z}^{eşik} = \begin{cases} 255, & I_{x,y,z} \geq i \\ 0, & I_{x,y,z} < i \end{cases} \quad (3.4.)$$

Bu eşitlikte  $I_{x,y,z}$ , eşikleme öncesindeki dekonvolve sinaptik veya segmente dendritik görüntülerin 3 boyutlu koordinatına  $(x, y, z)$  denk gelen sinyal intansite değerini;  $i$ , sinyal eşik değerini ve  $I_{x,y,z}^{eşik}$ , eşikleme sonrasında elde edilen görüntüyü simgelemektedir.



**Şekil 3.17.** Segmente dendrit ve dekonvole sinaptik sinyallerin yüksek geçirimli filtreleme ile eşiklemesi.

Ardından dendritten belirli bir mesafeye kadar olan sinaptik sinyallerin okunabilmesi için 3 katmanlı görüntü hazırlandı (Şekil 3.17. ve Şekil 3.18.). Bu işlem üç sinaptik sinyal türü için de ayrı yapıldı. Görüntüler hazırlanırken aşağıdaki işlemler uygulandı:

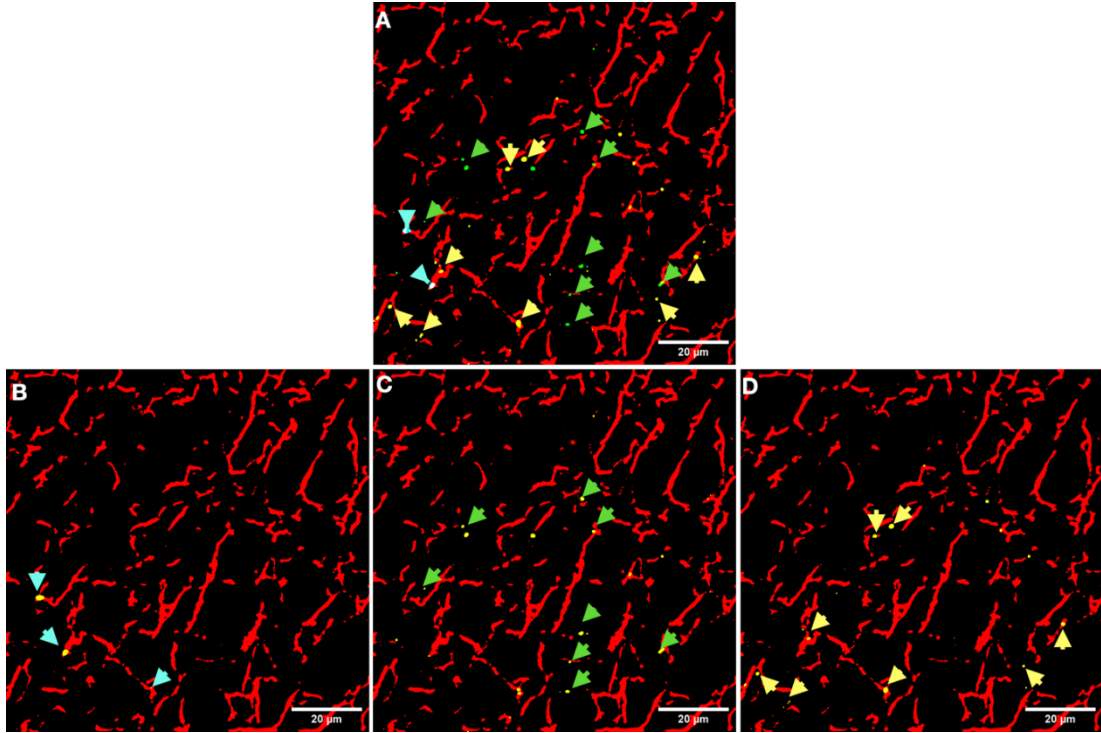
$$I = [I_R, I_G, I_B] \quad (3.5.)$$

$$I_R = I_{dendrit} \cup I_{sinaps} \quad (3.6.)$$

$$I_G = I_{sinaps} \quad (3.7.)$$

$$I_B = 0 \quad (3.8.)$$

Bu eşitliklerde  $I_{dendrit}$ , eşikleme sonrasında elde edilen 8-bit dendritik görüntü;  $I_{sinaps}$ , eşikleme sonrasında elde edilen 8-bit sinaptik görüntü;  $I$ , Synapse Quantifier algoritmasında işlenecek 3 boyutlu 8-bit RGB görüntü matrislerini simgelemektedir.  $I_R$ , dendritik ve sinaptik görüntü sinyal matrislerinin birleşiminden ibaret 3 boyutlu 8-bit matrisi;  $I_G$ , sinaptik görüntü sinyal matrisine denk 3 boyutlu 8-bit matrisi ve  $I_B$ , 3 boyutlu 8-bit sıfır matrisi simgelemektedir.



**Şekil 3.18.** Sinaptik sinyallerin eşikleme sonrası görüntüleri. Sinaptik sinyaller Synapse Quantifier algoritmasından renk renk geçirildi. Şekil 3.17.'deki filtrelenmiş görüntü renklerine ayrılarak gösterilmiştir.

### 3.10.3. Synapse Quantifier Uygulaması

Yukarıda da bahsedildiği üzere Synapse Quantifier algoritması sinyali tespit etmeden önce tespit edilecek en düşük hacim ve taranacak en uzun mesafe girdilerini kullanır. İncelediğimiz sinyallerin xy düzleminde Rayleigh çözünürlük limitleri 220-275 nm arasında değiştiği için incelenecek sinyallerin gürültüden ziyade sinaps sayılabilmesi için 3 düzlemde de en az 3 piksel =  $\sim 0.3 \mu\text{m}$  uzunluğa sahip olması gereklidir. Bu sebeple aranacak sinyallerin voksel eşiği en az 30 voksel olarak düzenlendi.

mPFK'de piramidal nöronlarının multifoton *in vivo* görüntülemesi, mPFK piramidal nöron dendritik çıkıntılarının dendrit daldan uzaklığının  $2 \mu\text{m}$  mesafeyi aşmadığını göstermiştir (114). Her ne kadar dar bir alandan veri gösterse de multifoton mikroskopi dendritik çıkıntılarının *in vivo* şekilde incelenmesi için yeterli bir teknik olmasından ötürü bu değer geçerli sayılmıştır (73). Bu sebeple maksimum sinaps arama mesafesi 20 piksel =  $\sim 2.2 \mu\text{m}$  olarak düzenlendi.

Hem dendritik hem sinaptik görüntülere eşikleme uygulandığı için görüntüdeki tüm sinyallerin intansitesi 8 bit görüntü cinsinden 255'e eşitlendiği için girilen dendritik ve sinaptik sinyal intansite eşik değerlerinin herhangi bir önemi kalmamıştır.

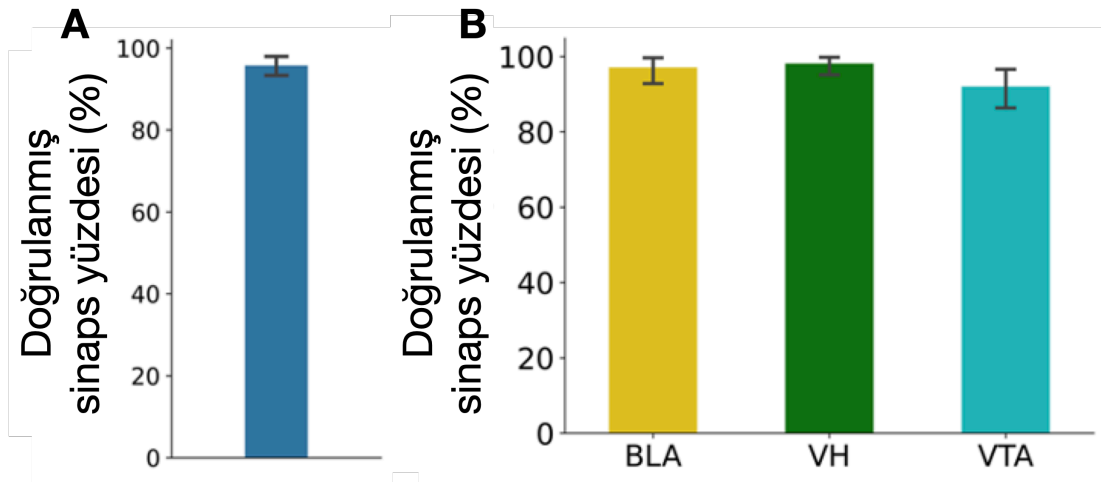
Bu projede büyük veriyle çalışıldığı ve veriyi oluşturan toplam insan sayısı "Bir" / "1" olduğu için sinapslar algoritma tespit edildikten sonra tek tek incelenmeden doğrudan kaydedilmiştir.

Synapse Quantifier algoritma çıktıları '.csv' uzantılı bir dosyaya kaydedilmiştir. Bu dosya içinde sinapsların 3 boyutlu koordinatları, sinapsın voksel cinsinden hacmi, dendrit ağacındaki en yakın olduğu nod ve dendritik çıkıntı üzerinde olup olmadığı bilgisi bulunur. Dendritik sinyal incelemesi esnasında büyük hacimli dendritik çıkıntıların gözle ayırt edilmesine rağmen, küçük hacimli dendritik çıkıntıların bu şekilde gözlenememesi üzerine Synapse Quantifier çıktı verisindeki sinapsın dendrit üzerindeki konum bilgisi (dendrit çıkıntı üzerinde olup olmadığı) analizler sırasında dikkate alınmamıştır.

#### **3.10.4. Synapse Quantifier Algoritmasının Elle Sayımla Karşılaştırılması**

Her ne kadar otomatize algoritmaların güvenilirliği birçok kez gösterilse de incelenen sinaptik sinyallerin niteliği sunulan otomatik sinaps tespit algoritmasında kullanılan veri setinden farklı olabileceği için incelenen sinapsların en azından bir kısmının elle sayım teknikleriyle karşılaştırılarak otomatik algoritma ile elle sayım sonuçlarının birbirlerine yakın olduğunun gösterilmesi gereklidir (142).

Bunun için önce her bir kortikal kolondan 600 x 2000 x 300 boyutlu rastgele 3 boyutlu görüntüler seçildi (n = 49). Önce, bu görüntülerde tespit edilen sinapslar, spektral ayrıştırılmış sinaps sinyalleriyle karşılaştırılarak doğrulandı. Bu işlem sonucunda tespit edilen sinapsların %95,7'sinin gerçek sinaps sinyalleriyle çakıştığı gözlemlendi (Şekil 3.19.).



**Şekil 3.19.** Örnek seçilmiş kortikal kolon bölgelerinde doğrulanmış, otomatik tespit edilen sinaps yüzdeleri.

**A:** Tüm sinapslar, **B:** Farklı beyin bölgeleri açısından doğrulanmış sinaps yüzdeleri.

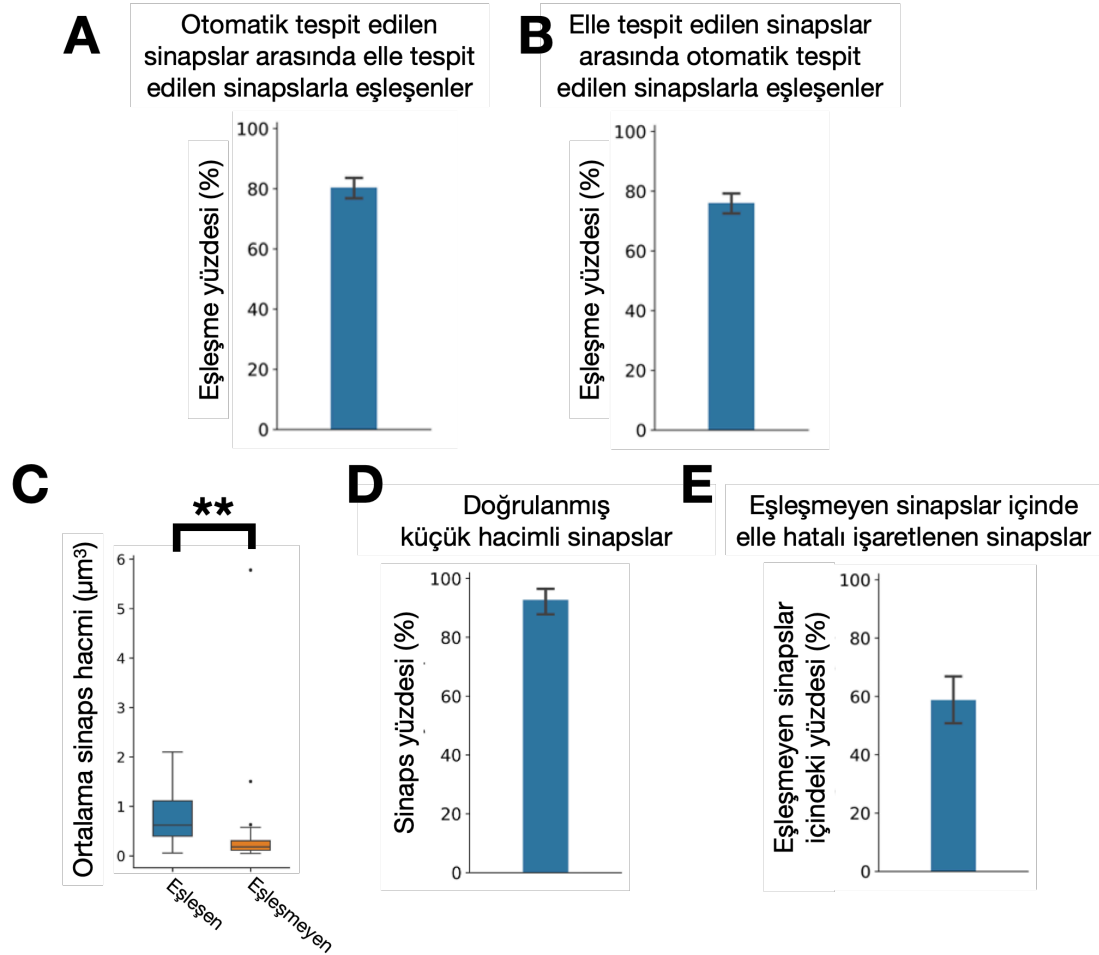
Sonraki aşamada, önce 3 boyutlu sinaps sinyalleri Vaa3D içerisinde 3 boyutlu görüntüleme modülü kullanılarak elle sayıldı. Synapse Quantifier dendritik rekonstrüksiyondan 20 piksel ( $\sim 2,2 \mu\text{m}$ ) uzaktaki sinyalleri incelediği için, elle sayım sırasında rekonstrüksiyon ağacından 20 piksel uzaktaki sinaptik sinyaller incelendi. İnceleme sonrasında sinapsların 3 boyuttaki koordinatları *.marker* uzantılı bir dosyaya kaydedildi.

Elle ve otomatik sayılan birbirlerine en yakın sinapslar Macar algoritmasıyla (*Hungarian algorithm*) uzaysal şekilde eşleştirildi (143). Sonrasında eşleşen sinapslar arasındaki Öklid mesafesi hesaplandı. Öklid mesafesi  $2 \mu\text{m}$ 'den daha kısa olan sinaps çiftleri uzaysal eşleşmiş sayıldı.

Sonuçta önce elle tespit edilen sinapsların ortalama %76'sının otomatik sinaps sayımıyla tespit edilen sinapslarla eşleştiği tespit edildi. Ancak otomatik tespit edilen sinapsların da %20'sinin de elle sayım sırasında tespit edilemediği gözlemlendi. Elle sayım sırasında tespit edilemeyen bu sinapslar yakından incelendiğinde, bu sinapsların hacimleri otomatik sinaps sayımıyla eşleşen sinapsların hacimlerinden anlamlı şekilde daha düşük olduğu gözlemlendi. Yine otomatik ve elle sayımda eşleşen sinapslar retrospektif olarak ham görüntülerde incelendiğinde bu sinapsların %93'ünün gerçekten sinaptik sinyal olduğu gözlemlendi. Son olarak elle tespit edilip, otomatik tespit edilemeyen sinapsların %58,7'sinin herhangi bir sinaptik sinyal içermediği gözlemlendi (Şekil 3.20.).

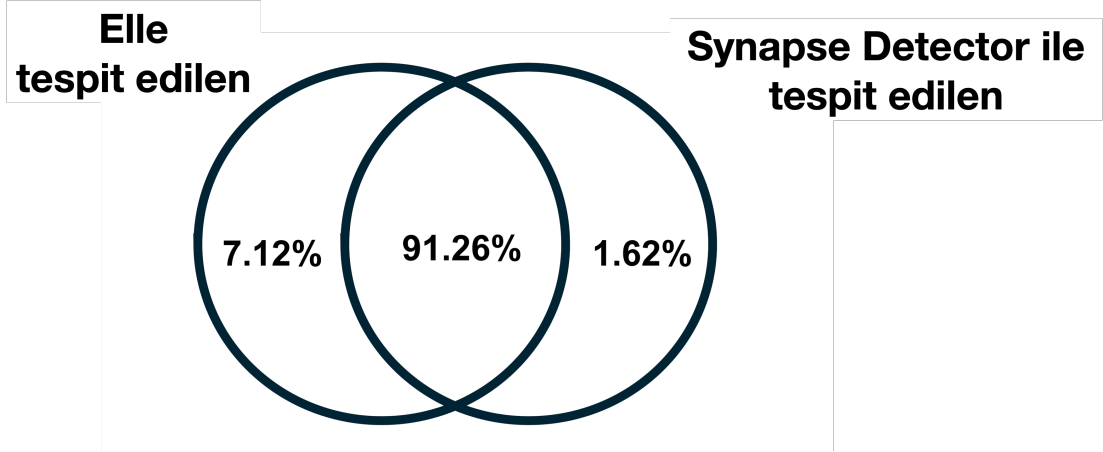


Bu durum elle sayım sırasında Vaa3D 3 boyut görüntüleme modülünde yapılan elle işaretlerin kaymasıyla ilişkilendirilmiştir.



**Şekil 3.20.** Otomatik ve elle tespit edilen sinapsların uzaysal ve hacimsel karşılaştırması. C için  $p = 2.57 \times 10^{-8}$ , Mann-Whitney U testi.

Sonuçta, elle sayımda sinaps olmayan koordinatların dışlanması ve otomatik sayımda görülen küçük hacimli sinapsların dahil edilmesi sonrasında %91,26 oranda otomatik sinaps işaretleme performansı elde edilmiştir (Şekil 3.21.).



Şekil 3.21. Synapse Detector algoritmasının genel performansının özeti.

### 3.11. Kortikal Katmanların Tanımlanması ve Rekonstrüksiyonlarının Hazırlanması

Kortikal katmanlar Allen Beyin Atlası'nda bulunan *Common Coordinate Framework (CCF)* içerisine mPFK için tanımlanmış katman sınırları dikkate alınarak hesaplandı (40). İnceleme sonucunda 1. katman, 2/3. katman, 5. katman ve 6. katmanın kortikal kolunun sırasıyla %8, %14, %61 ve %17'sini barındıracağı hesaplandı. Ayrıca 5. katman, literatürdekine benzer biçimde 5a katmanı ve 5b katmanı şeklinde ikiye ayrıldı. Sonrasında 3 boyutlu kortikal rekonstrüksiyonlar bu katmanlara horizontal şekilde kesilerek ayrıldı. İki katmandan geçen segmentler kortikal katmanların incelemesi sırasında analizin dışında bırakıldı.

### 3.12. Her Bölge için Sinaps Yüzde Hesabı

Kortikal kolon katmanlarına ayrıldıktan sonra, her bir beyin bölgesi için o beyin bölgesinden gelen sinaps sayısının, işaretlenmiş toplam sinaps sayısına oranı hesaplandı.  $n_{katman}^{renk}$ , 1., 2/3., 5a, 5b veya 6. katmanlardaki BLA, VH veya VTA kaynaklı sinaps sayısı iken

$$N_{katman} = n_{katman}^{camgöbeği} + n_{katman}^{yeşil} + n_{katman}^{sarı} \quad (3.9.)$$

$$P_{katman}^{renk} = \frac{n_{katman}^{renk}}{N_{katman}} \times 100; renk \in \{camgöbeği, yeşil, sarı\} \quad (3.10.)$$

Bu eşitliklerde  $N_{katman}$ , bir katmanda tespit edilen toplam sinaps sayısını;  $P_{katman}^{renk}$ , bir katmanda o renge sahip (bir beyin bölgesinden kaynaklanan) sinaps yüzdesini ifade etmektedir. Ardından sonraki işlemlerde kullanılacak renk yüzde matrisi ( $M^{renk}$ ) oluşturuldu:

$$M_{i,katman}^{renk} = P_{katman}^{renk} \quad (3.11.)$$

$M_{i,katman}^{renk}$ , camgöbeği, yeşil ve sarı eGRASP ile işaretli sinapsların  $i$ , kortikal kolonun bir katmanındaki yüzdesini simgelemektedir.  $i$  kortikal kolonun kimliğini göstermektedir. Bu işlem sonrasında her bir pre-sinaptik girdi bölgesi için 5 x 13 boyutunda bir matris elde edildi.

### 3.13. Boyut İndirgeme (*Dimensionality Reduction*)

Boyut indirgeme işlemi önce  $M^{renk}$  matrisleri sıralı şekilde birleştirildi.

$$M^P = [M^{camgöbeği}, M^{yeşil}, M^{sarı}] \quad (3.12.)$$

Böylece son boyutu 13 x 15 olan boyut indirgeme matrisi ( $M^P$ ) elde edildi. Ardından temel bileşen analizi (*Principal Component Analysis, TBA*) uygulandı. 1. ve 2. temel bileşenler (*principal components, TB*) sinaps sayı yüzdelерinin düşük boyuttaki projeksiyonu için kullanıldı.

### 3.14. BLA ve VH Sinapsları İçin En Yakın Uzaysal Mesafe Matrislerin Oluşturulması ve İncelenmesi

5b ve 6. katmandaki BLA ve VH sinapslarının uzaysal en yakın ilk 4'er BLA ve VH sinapsa mesafeleri ölçüldü. Sonrasında bu mesafelerin her bir sinaps için sıralanmasıyla 42,317 x 8 boyutlu bir matris elde edildi. Bu matrisi ilk 4 kolunu en yakın ilk 4 VH sinapsına uzaklıklarını, diğer 4 kolunu da en yakın ilk 4 BLA sinapsına uzaklıklarını içermektedir. Sonra *Uniform Manifold Approximation and Projection (UMAP)* yöntemi sinapsların, davranışsal fenotiplerin, sosyal etkileşim skorlarının ve fare kimliklerinin görüntülemesi için kullanıldı (144). Bu işlem sırasından UMAP kod kitabındaki varsayılan değişken hiperparametreleri aynen kullanıldı.

Rastgele orman sınıflayıcılar (*random forest classifiers*) için 75'er ağaçlık 10 orman kullanıldı. Her ormanda 10-katmanlı çapraz doğrulama (*k-fold cross validation*) uygulandı. Bu süreçte verinin %10'u test için kullanıldı. Rastgele dağılımla karşılaştırma için verilerin etiketleri rastgele şekilde değiştirildi. Parametrik analizler, doğruluk (*accuracy*) ve F1 skorları çalışma verisi ve rastgele verinin karşılaştırılması için kullanıldı. Boyut indirgeme ve rastgele orman sınıflama işlemleri Python 3.9 içerisinde bulunan sklearn v1.4.0 kütüphanesi kullanılarak yapıldı.

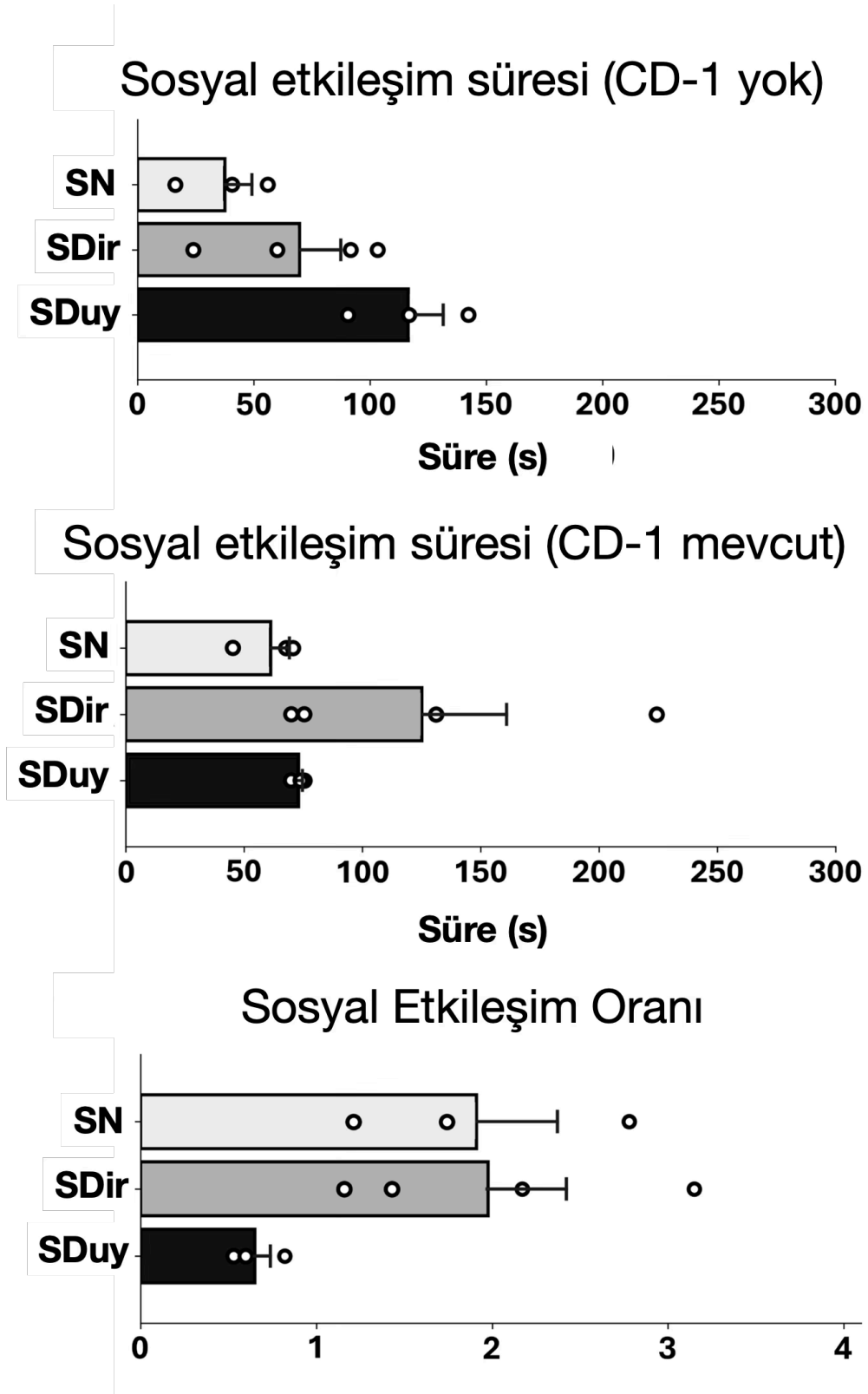
### 3.15. İstatistik Analiz

Veriler medyan ve çeyrekler arası açıklık, ortalama  $\pm$  standart ortalama hata veya ortalama ve %95 güven aralığı şekillerinde sunuldu. Kortikal kolon karşılaştırmasında grup sayıları düşük olduğu için non-parametrik testler karşılaştırma amaçlı kullanıldı: Kruskal-Wallis test çok gruplu karşılaştırma amaçlı; Mann-Whitney U test ve Bonferroni düzeltmesi ikili grup karşılaştırmaları amaçlı kullanıldı (145). Tüm analizler iki yönlü şekilde yapıldı. p değerleri Kruskal-Wallis testinde 0,05'ten düşük olduğunda; *post-hoc* Mann-Whitney U testinde 0,0167'den düşük olduğunda istatistiksel anlamlı kabul edildi. İstatistik analizler Python 3.9 içerisinde bulunan Scipy v1.11.1 kütüphanesi kullanılarak gerçekleştirildi.

## 4. BULGULAR

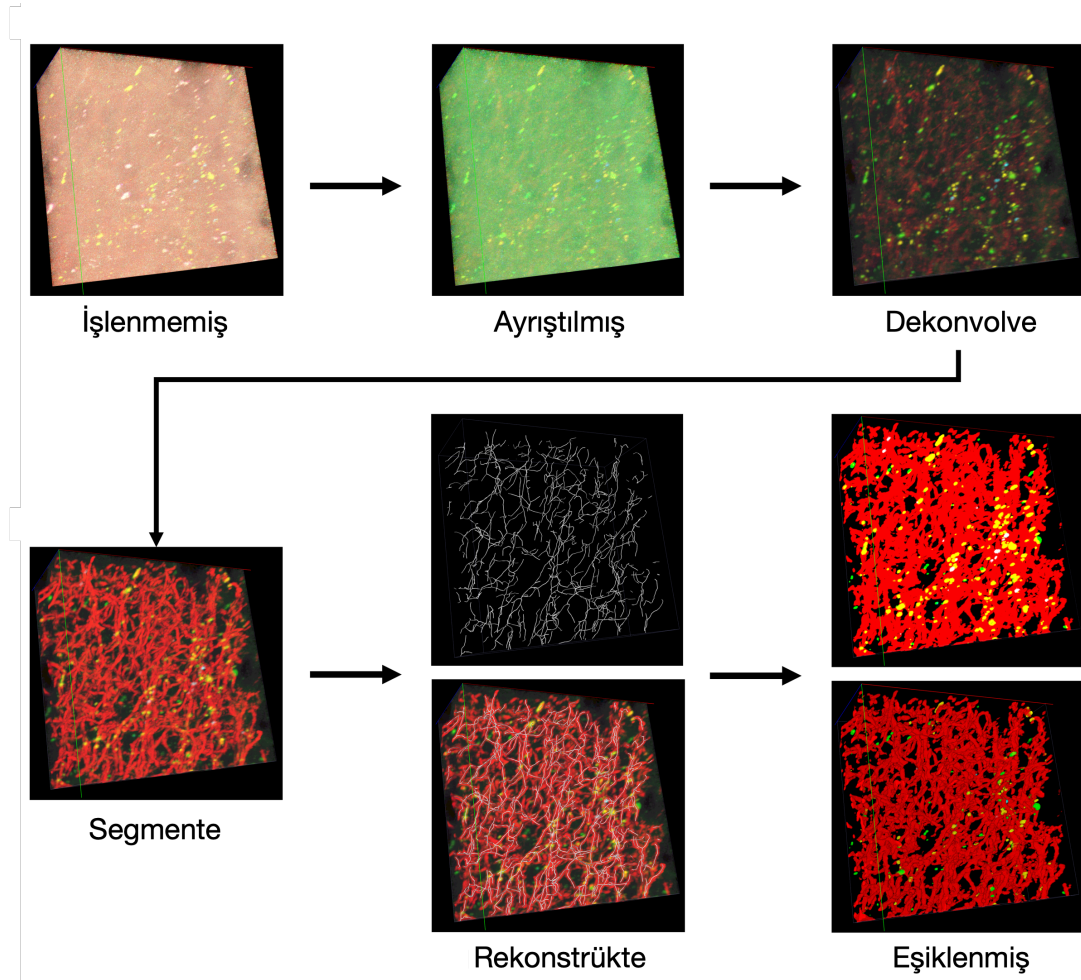
### 4.1. mPFK piramidal nöron dendritlerinde BLA, VH ve VTA piramidal nöronlarından kaynaklı girdiler

BLA, VH ve VTA kaynaklı mPFK sinapslarının işaretlenmesi için sarı, yeşil ve camgöbeği pre-eGRASP ve post-eGRASP ifade eden AAV'ler BLA, VH, VTA ve mPFK'ye enjekte edildi. Stres grubu fareler 3 hafta sonra 10 günlük kronik sosyal yenilgi stresine (KSYS) maruz bırakıldı. Stres-naif fareler kendisiyle aynı cinsten bir kafes arkadaşıyla saydam bir ayraçla ayrılacak şekilde olmak kaydıyla aynı kafese yerleştirildi. 11. günde sosyal etkileşim testiyle stres grubu fareler stres duyarlı (SDuy) (n = 3) ve stres dirençli (SDir) (n = 4) olmak üzere ikiye ayrıldılar. Stres naif (SN) farelerin bu test sonrasında stres dirençli (ilk kez gördüğü CD-1 fareyle daha fazla vakit geçirdiği) olduğu görüldü (Şekil 4.1.). Doku hazırlığından sonra mPFK'nin yüzeyinden derinine multispektral, yüksek çözünürlüklü 3 boyutlu görüntüler kiremit tarama yöntemiyle alındı (~200 x 1500 x 30  $\mu\text{m}^3$ ).

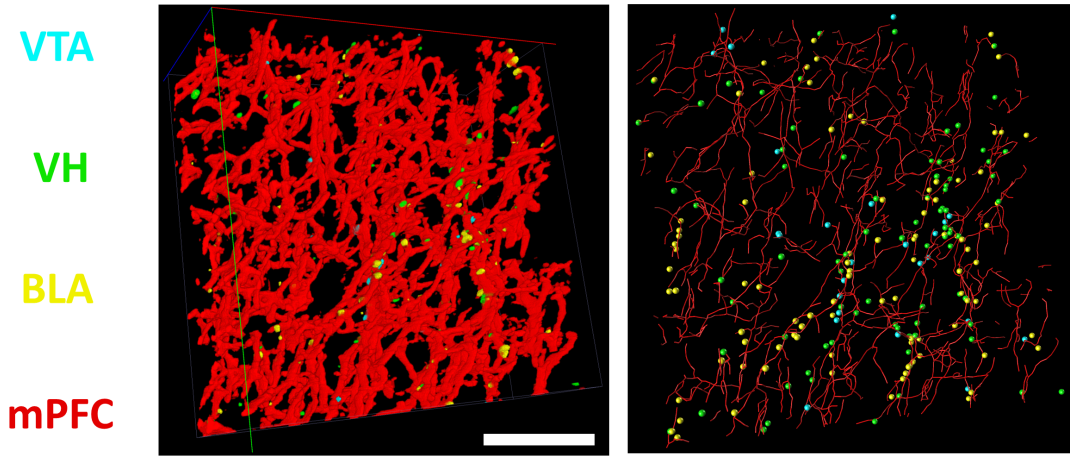


**Şekil 4.1.** Çalışmada incelenen farelerin CD-1 fare sosyal etkileşim bölgesinde bulunmadığı, bulunduğu dönemdeki sosyal etkileşim süreleri ve sosyal etkileşim oranları.

Çalışmada incelenen fluoroforların emisyon dalga boyu sinyali eğrileri çakıştığı için sinaptik ve dendritik sinyalleri *Learning Unsupervised Means of Spectra (LUMoS)* algoritmasıyla ayrıştırıldı (127). Sonrasında dendritik ve sinaptik sinyaller dekonvolve edildi. Ardından dendritik sinyaller yapay sinir ağı (*artificial neural network*) tabanlı bir modelle segmente edildi ve 3 boyutlu dendritik ağacın eldesi için rekonstrükte edildi (136). Bu süreçte dendritik ağacın ve dekonvolve dendritik sinyalin başarılı şekilde çakıştığı gözlemlendi. İncelenen kortikal hacmi çok büyük olduğundan dolayı elle sinaps işaretleme sayım yapımı oldukça zaman maliyetlidir. Bu sebeple sinapsların otomatik tespiti için Synapse Detector (140) algoritması kullanıldı (Şekil 4.2. ve Şekil 4.3.).



Şekil 4.2. Örnek görüntüde görüntü işleme basamakları.



**Şekil 4.3.** Synapse Detector algoritmasının örnek sonucu. Ölçek: 20  $\mu$ m.

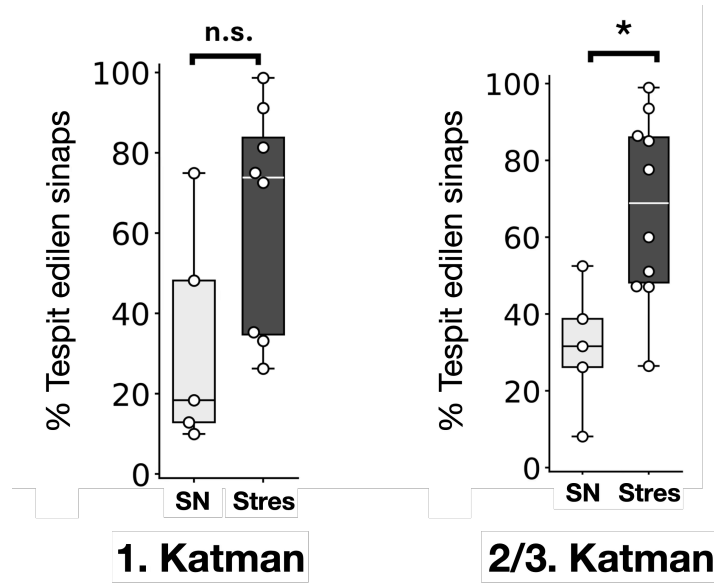
İşlemler sonucunda tespit edilen sinapsların medyanı SN, SDir, SDuy fareler için sırasıyla, 2,501 (çeyrekler açıklığı (ÇA) = 1.077 – 6.293, n = 3 fare); 7,509 (ÇA = 2.560 – 9.177; n = 4 fare) ve 5.845 (ÇA = 789 – 8.598; n=3 fare) olduğu görüldü. SN grupta, tespit edilen sinapsların büyük bir çoğunluğunun VH-mPFC bağlantına ait olduğu görüldü (%70,33  $\pm$  16,82). Bu sinapsları sırasıyla BLA-mPFC (%16,09  $\pm$  11,25) ve VTA-mPFC (%13,57  $\pm$  6,09) sinapsları takip etti. SDir grupta VH-mPFC (%49,62  $\pm$  17,80) ve BLA-mPFC (%38,17  $\pm$  20,37) sinapslarının oranları birbirlerine benzer görüldü (p = 0,69, Mann-Whitney U testi (MWU)). VTA-mPFC sinapslarının oranı daha düşüktü (%12,19  $\pm$  4,32). SDuy grupta BLA-mPFC sinapsların oranı, VH-mPFC sinapsların oranından anlamlıya yakın eğilim gösterecek şekilde daha yüksek gözlemlendi (sırasıyla, %54,27  $\pm$  17,18 ve 25,74  $\pm$  15,12%; p = 0,055; MWU testi). SDuy grupta VTA-mPFC sinapslarının oranının (%19,97  $\pm$  9,79) diğer gruplarla benzer olduğu gözlemlendi (p = 0,30; MWU). Tespit edilen BLA-mPFC, VH-mPFC ve VTA-mPFC sinapslarının oranları SDir ve SDuy gruplarda benzer olduğu gözlemlendi (sırasıyla, p = 0,09; p = 0,15 ve p = 0,69; MWU testi).

#### **4.2. BLA-mPFC sinaps oranlarında kortikal katman ve stres fenotipi bağımlı değişiklikler**

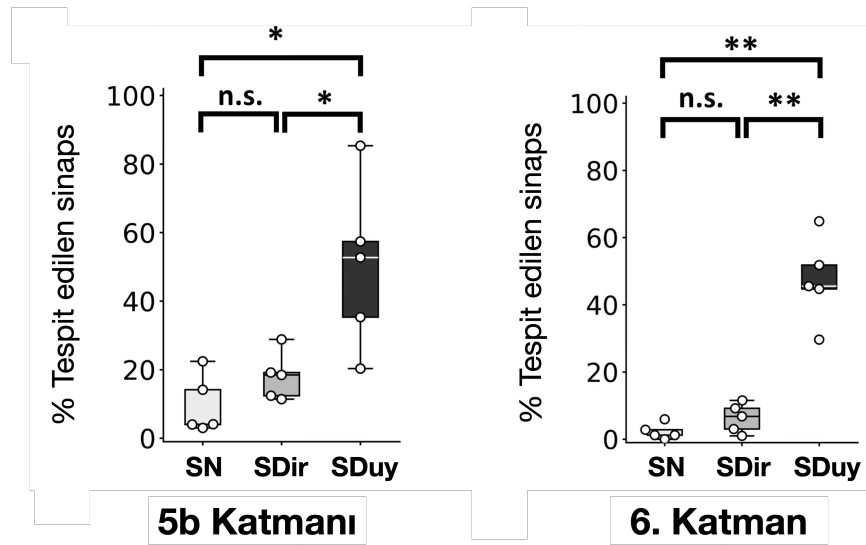
Sonra, her bir kortikal katmanda sinapsların dağılımı incelendi. Önceki literatürde BLA ve VH aksonal uzanımları mPFC'nin sırasıyla 2/3. katmanı ve 5. katmanını hedeflediği gözlemlenmiştir (10, 24-26). Bu verilerle uyumlu olacak şekilde, tespit edilen SN farelerde BLA-mPFC sinapslarının tüm tespit edilen sinapslara olan



yüzdesinin 2/3. katmanda 5a katmanı ve 6. katmana göre anlamlı şekilde daha fazla olduğu gözlemlendi (5a ve 6 için sırasıyla,  $p = 0,0079$ ;  $p = 0,031$ ; MWU test). Kronik strese maruz bırakılan farelerde 2/3. katmandaki BLA-mPFC sinapsların oranı, SN farelere kıyasla anlamlı şekilde artmıştı ( $p = 0,019$ , MWU testi) (Şekil 4.4.). Ancak bu oranın 5b katmanı ve 6. katmanda sadece SDuy farelerde SDir ve SN farelere göre anlamlı şekilde daha yüksek olduğu görüldü (Şekil 4.5.). 5b ve 6. katmanlarındaki BLA sinapslarının yüzdesiyle sosyal etkileşim oranı arasında anlamlı negatif korelasyon vardı (Şekil 4.6.).

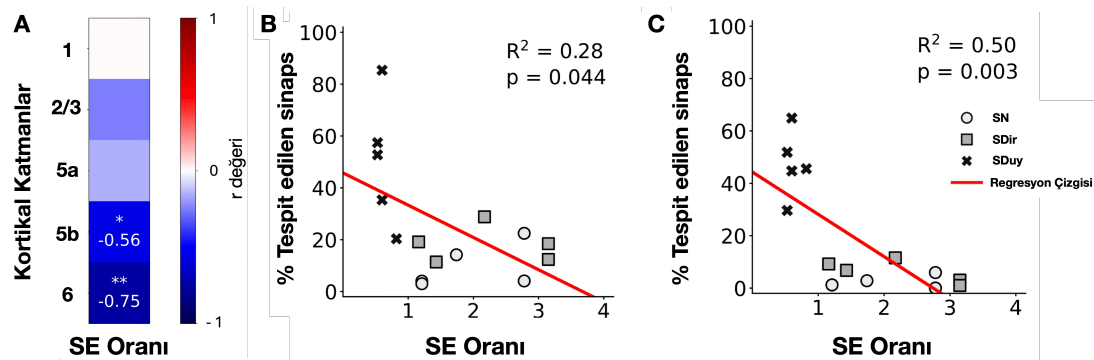


**Şekil 4.4.** 1. katman ve 2/3. katmanda BLA-mPFC sinapslarının tüm tespit edilen sinapslara oranının karşılaştırması. \*:  $p < 0,05$ 'i simgelemektedir.



**Şekil 4.5.** 5b katmanında ve 6. katmanda BLA-mPFK sinapslarının tüm tespit edilen sinapslara oranının karşılaştırması.

5b katmanı için,  $p = 0,012$ ,  $H = 8,78$ , Kruskal-Wallis testi; SDir-SDuy için  $p = 0,0158$ ; SN-SDuy için  $p = 0,0158$  ve SN-SDir için  $p = 0,22$ , MWU testi. 6. katman için  $p = 0,0052$ ,  $H = 10,50$ , Kruskal-Wallis testi; SDir-SDuy için  $p = 0,0079$ , SN-SDuy için  $p = 0,0079$ , SN-SDir için  $p = 0,15$ , MWU testi. \*:  $p < 0,05$  ve \*\*:  $p < 0,01$ 'i simgelemektedir.

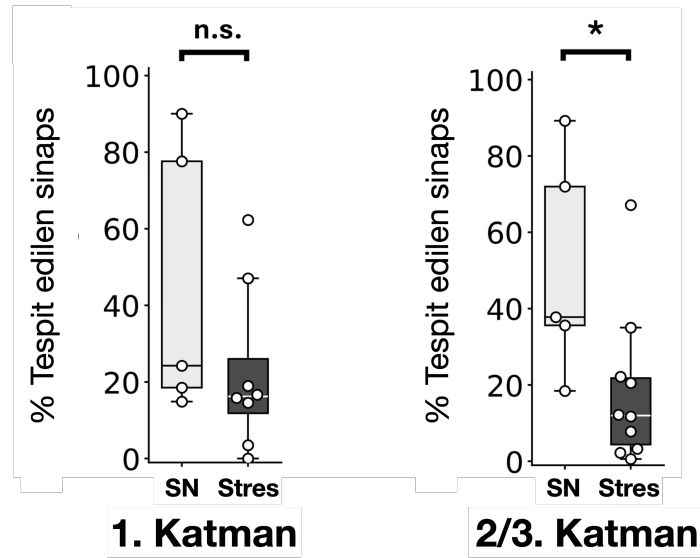


**Şekil 4.6.** 5b katmanında ve 6. katmanda BLA-mPFK sinapslarının tüm tespit edilen sinapslara oranlarının sosyal etkileşim oranıyla korelasyonları.

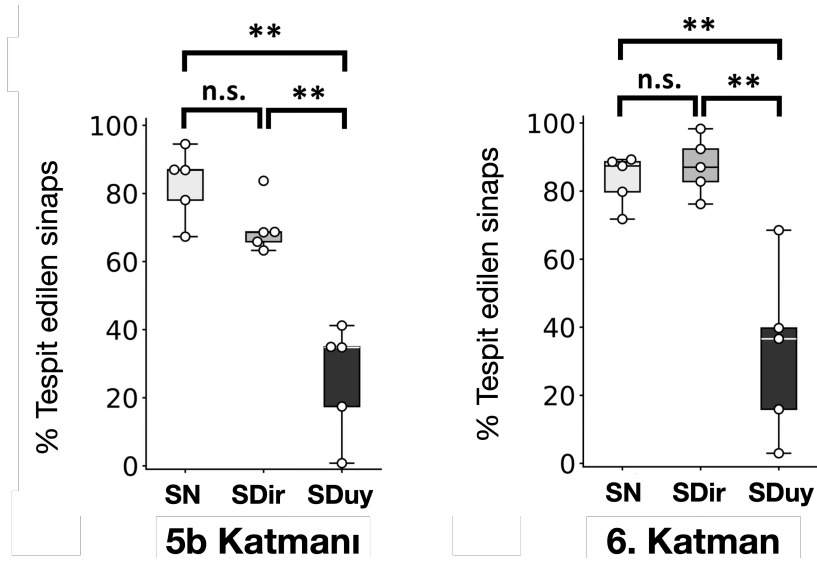
**A)** Spearman korelasyonların derinlik bağımlı şekilde renk koduyla gösterimi, **B)** 5b katmanı ve **C)** 6. katmandaki korelasyon sonuçları. Lineer regresyon katsayıları ve p değerleri şekil üzerinde gösterilmiştir. \*:  $p < 0,05$  ve \*\*:  $p < 0,01$ 'i simgelemektedir.

### 4.3. VH-mPFK sinaps oranlarında kortikal katman ve stres fenotipi bağımlı değişiklikler

VH-mPFK sinapslarında gözlemlenen değişiklikler BLA-mPFK ile benzer şekilde ancak zıt yöndeydi. Kronik strese maruz kalma sonucu VH-mPFK sinapslarının tüm tespit edilen sinapslara yüzdesi 2/3. katmanda anlamlı şekilde azaldı ( $p = 0,019$ , MWU test) (Şekil 4.7.). Bu katmanda, VH-mPFK sinaps yüzdesi SDir ve SDuy fareler arasında herhangi bir fark göstermedi. Ancak 5b katmanında ve 6. katmanda VH-mPFK sinaps yüzdesi SDuy farelerde SN ve SDir farelere göre anlamlı şekilde düşüktü (Şekil 4.8.). Aynı katmanlarda VH-mPFK sinapslarının yüzdeleri ile sosyal etkileşim oranları arasında anlamlı korelasyon gözlemlendi (Şekil 4.9.).

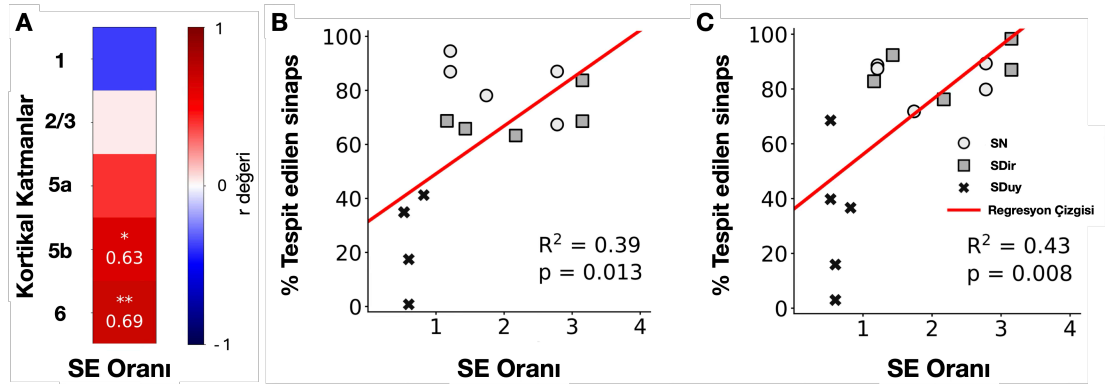


Şekil 4.7. 1. katman ve 2/3. katmanda VH-mPFK sinapslarının tüm tespit edilen sinapslara oranının karşılaştırması. \*:  $p < 0,05$ 'i simgelemektedir.



**Şekil 4.8.** 5b katmanında ve 6. katmanda VH-mPFK sinapslarının tüm tespit edilen sinaplara oranının karşılaştırması.

5b katmanı için:  $p = 0,0044$ ,  $H = 10,82$ , Kruskal-Wallis testi; SDuy-SDir için  $p = 0,0079$ , SS vs SR; SN-SDuy için  $p = 0,0079$ ; SN-SDir için  $p = 0,09$ , MWU testi. 6. katman için: KW test:  $p = 0,0086$ ,  $H = 9,50$ , Kruskal-Wallis testi; SDuy-SDir için  $p = 0,0079$ , SN-SDuy için  $p = 0,0079$ , SN-SDir için  $p = 0,69$ , MWU testi. \*:  $p < 0,05$  ve \*\*:  $p < 0,01$ 'i simgelemektedir.

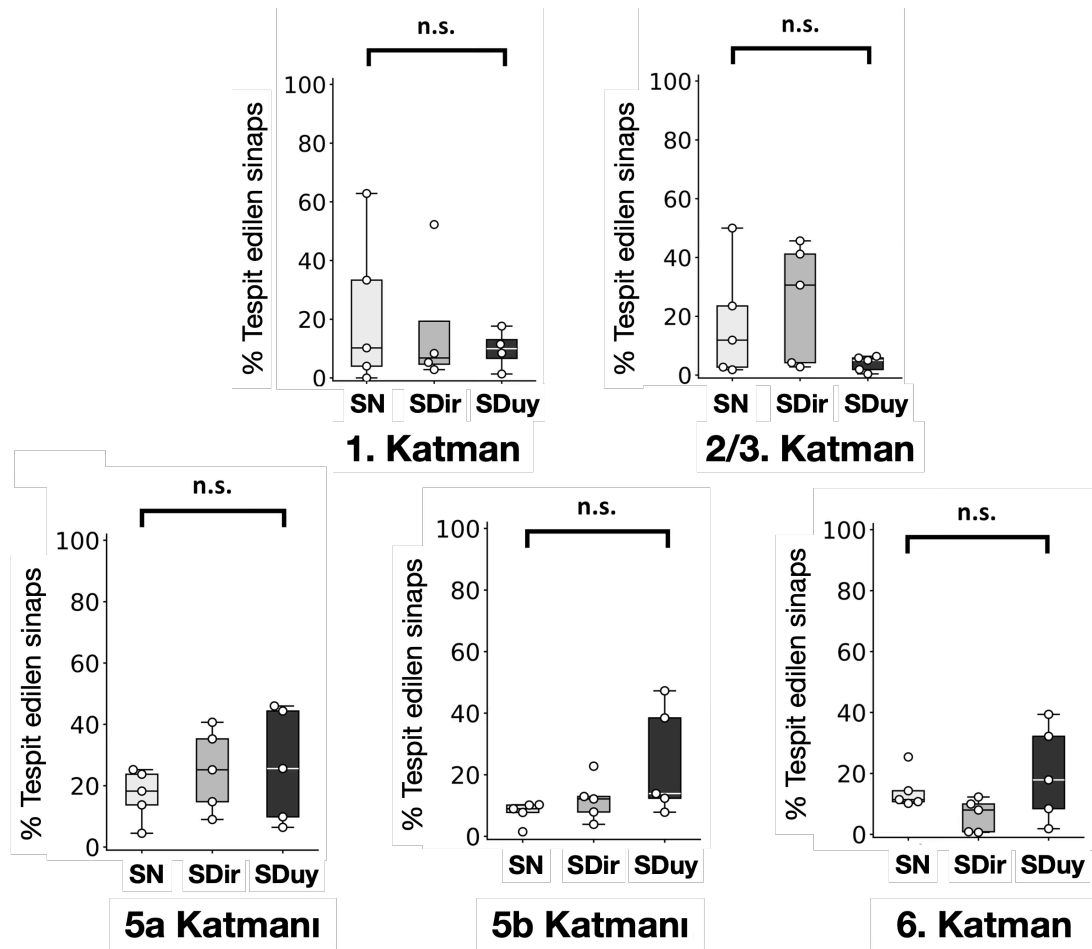


**Şekil 4.9.** 5b katmanında ve 6. katmanda VH-mPFK sinapslarının tüm tespit edilen sinaplara oranlarının sosyal etkileşim oranıyla korelasyonları.

A) Spearman korelasyonların derinlik bağımlı şekilde renk koduyla gösterimi, B) 5b katmanı ve C) 6. katmandaki korelasyon sonuçları. Lineer regresyon katsayıları ve p değerleri şekil üzerinde gösterilmiştir. \*:  $p < 0,05$  ve \*\*:  $p < 0,01$ 'i simgelemektedir.

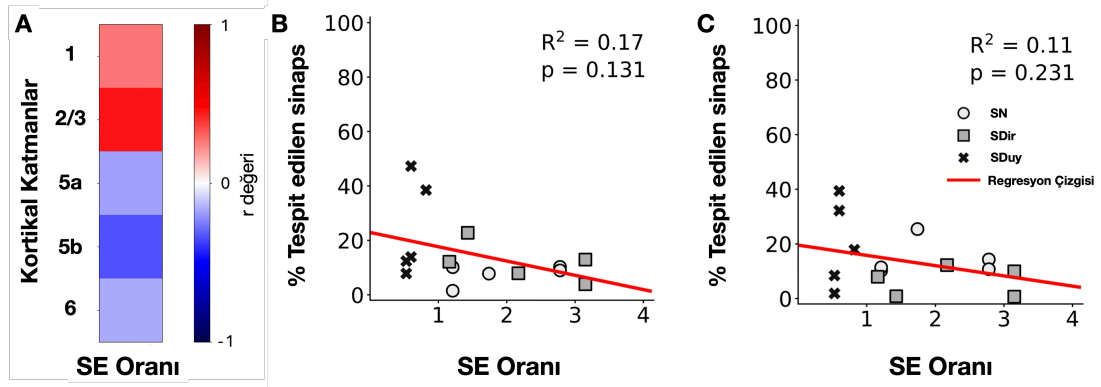
#### 4.4. VTA-mPFK sinaps oranlarında kortikal katman ve stres fenotipi bağımlı değişiklikler

Tüm deney gruplarında, VTA-mPFK sinapslarının tüm tespit edilen sinapslara yüzdesi tüm katmanlarda benzer şekilde gözlemlendi (Şekil 4.10.). VTA-mPFK sinapslarıyla sosyal etkileşim oranları arasında anlamlı ilişki saptanmadı (Şekil 4.11.).



**Şekil 4.10.** Tüm katmanlarda VTA-mPFK sinapslarının tüm tespit edilen sinapslara oranının karşılaştırması.

1. katman için,  $p = 0,94$ ,  $H = 0,11$ ; 2/3. katman için,  $p = 0,28$ ,  $H = 2,47$ ); 5a katmanı için  $p = 0,53$ ,  $H = 1,26$ ); 5b katmanı için,  $p = 0,10$ ,  $H = 4,5$  ve 6. katman için,  $p = 0,11$ ,  $H = 4,34$ , Kruskal-Wallis testi.

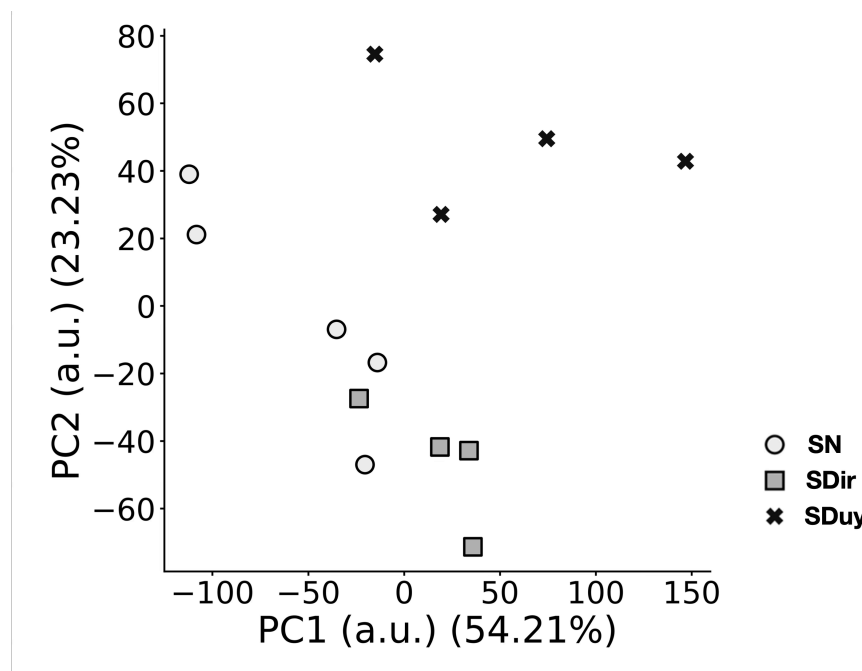


**Şekil 4.11.** 5b katmanında ve 6. katmanda VTA-mPFK sinapslarının tüm tespit edilen sinaplara oranlarının sosyal etkileşim oranıyla korelasyonları.

**A)** Spearman korelasyonların derinlik bağımlı şekilde renk koduyla gösterimi, **B)** 5b katmanı ve **C)** 6. katmandaki korelasyon sonuçları. Lineer regresyon katsayıları ve p değerleri şekil üzerinde gösterilmiştir.

#### 4.5. BLA-mPFC, VH-mPFC ve VTA-mPFC sinapslarının kortikal kolon boyunca görelî miktarlarının düşük boyuttaki temsilleriyle SDuy farelerin SDir farelerden ayırt edilmesi

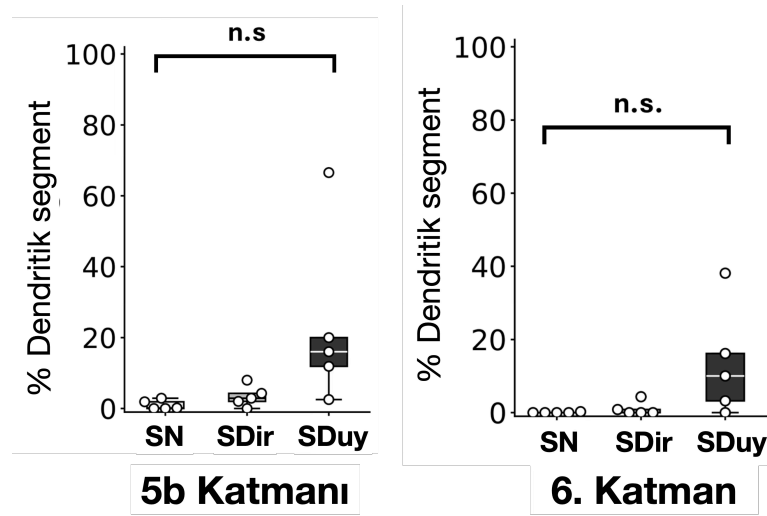
Grupların strese duyarlılığının daha fazla incelenmesi için sinaps oranlarının düşük boyuttaki yapıları, temel bileşen analiziyle (*principal component analysis, TBA*) 13 kortikal kolonda incelendi. Bunun için önce 3x13x5'lik (bölge sayısı x kolon sayısı x katman sayısı) sinaps oran matrisi aynı kolonda her rengin bilgisi peş peşe dizilerek 13 x 15'lik bir matrise dönüştürüldü. Sonrasında TBA uygulanıp kortikal kolondaki sinaps oranları ilk 2 temel bileşen (*principal component, TB*) ekseninde yerleştirilerek 2 boyutta görüntüldü. Bu iki TB, sinaptik oranların değışkenliğinin %77,44'ünü yakalayabildi. Kortikal kolonlar bu iki TB düzleminden SDuy farelerin SDir farelerden farklı yerlerde bulunduğu gözlemlendi (Şekil 4.12.). Bu bulgu, BLA-mPFC, VH-mPFC ve VTA-mPFC kortikal kolon boyu sinapslarının oranlarının SDuy ve SDir fareler arasındaki farkı açıklayabileceğini işaret etmektedir.



Şekil 4.12. BLA-mPFC, VH-mPFC ve VTA-mPFC sinapslarının kortikal kolon boyu farklı kortikal katmanlardaki oranlarının TBA ile iki boyuta indirgenmiş koordinatları.

#### 4.6. Birden fazla bölgeden sinaptik girdi alan dendritlerin miktarının SDuy farelerde arttığı gözlemlendi

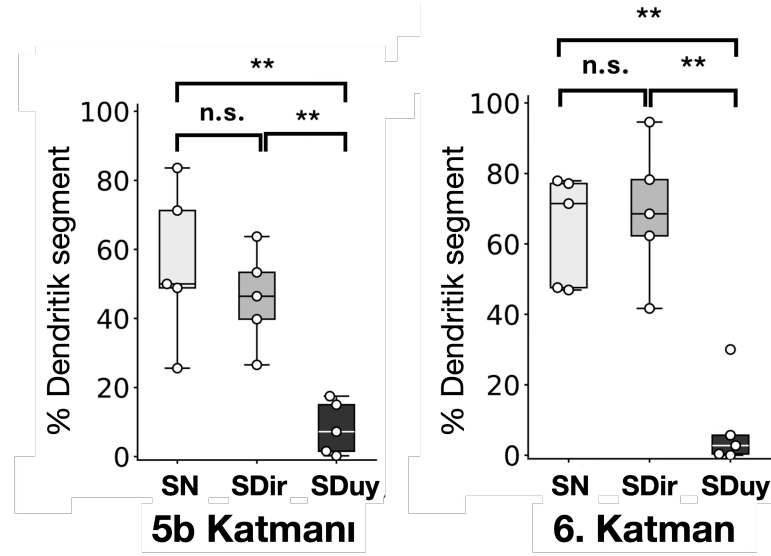
SDuy farelerin stres yanıtındaki mikrodevresel farklılıkların daha iyi anlaşılması için 5b ve 6. katmanlarındaki en az iki tane sinaptik girdi alan dendritik segmentler incelendi. SDuy farelerin hem 5b katmanı hem de 6. katmanlarında sadece BLA'dan sinaptik girdi alan segmentler anlamlılığa yakın şekilde daha yüksek; sadece VH'den girdi alan dendritik segment oranları diğer farelere göre anlamlı şekilde daha düşüktü (Şekil 4.13. ve Şekil 4.14.). Sadece VTA'dan sinaptik girdi alan segmentlerin oranları gruplar arasında anlamlı farklılık göstermedi (Şekil 4.15.).



**Şekil 4.13.** 5b katmanı ve 6. katmanda sadece BLA'dan sinaptik girdi alan dendritik segment oranlarının karşılaştırması.

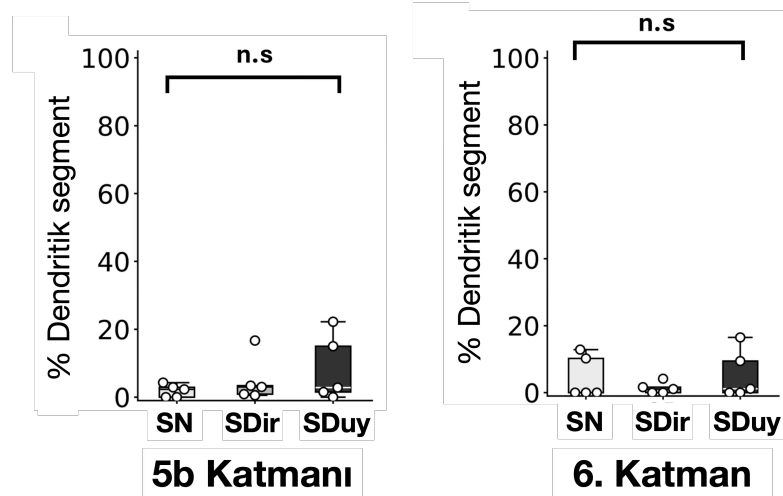
5b katmanı için,  $p = 0,020$ ,  $H = 7,79$ , Kruskal-Wallis testi; SDir-SDuy için  $p = 0,055$ , SN-SDuy için  $p = 0,021$  ve SN-SDir için  $p = 0,20$ , MWU testi; 6. katman için,  $p = 0,054$ ,  $H = 5,82$ , Kruskal-Wallis testi.





**Şekil 4.14.** 5b katmanı ve 6. katmanda sadece VH'den sinaptik girdi alan dendritik segment oranlarının karşılaştırması.

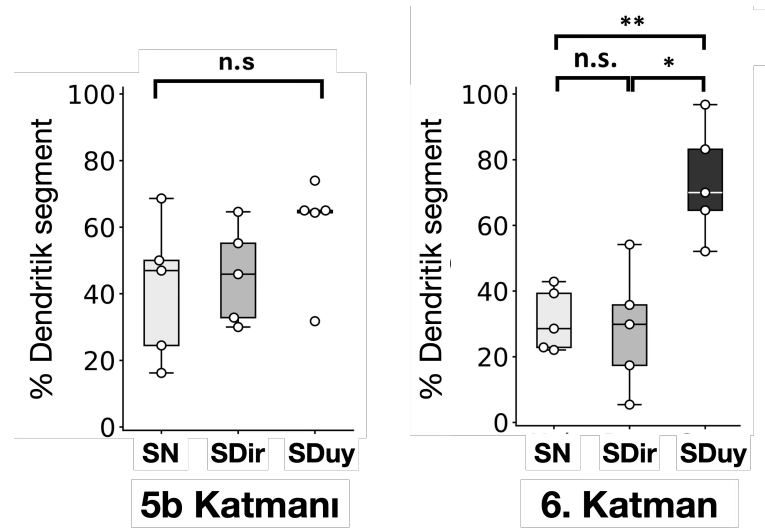
5b katmanı için,  $p = 0,0081$ ,  $H = 9,62$ , Kruskal-Wallis testi; SDuy-SDir için  $p = 0,0079$ , SN-SDuy için  $p = 0,0079$ , SN-SDir için  $p = 0,54$ , MWU testi. 6. katman için  $p = 0,0090$ ,  $H = 9,42$ , Kruskal-Wallis testi; SDuy-SDir için  $p = 0,0079$ , SN-SDuy için  $p = 0,0079$  ve SN-SDir için  $p = 0,84$ , MWU testi. \*\*:  $p < 0.01$ 'i simgelemektedir.



**Şekil 4.15.** 5b katmanı ve 6. katmanda sadece VTA'dan sinaptik girdi alan dendritik segment oranlarının karşılaştırması.

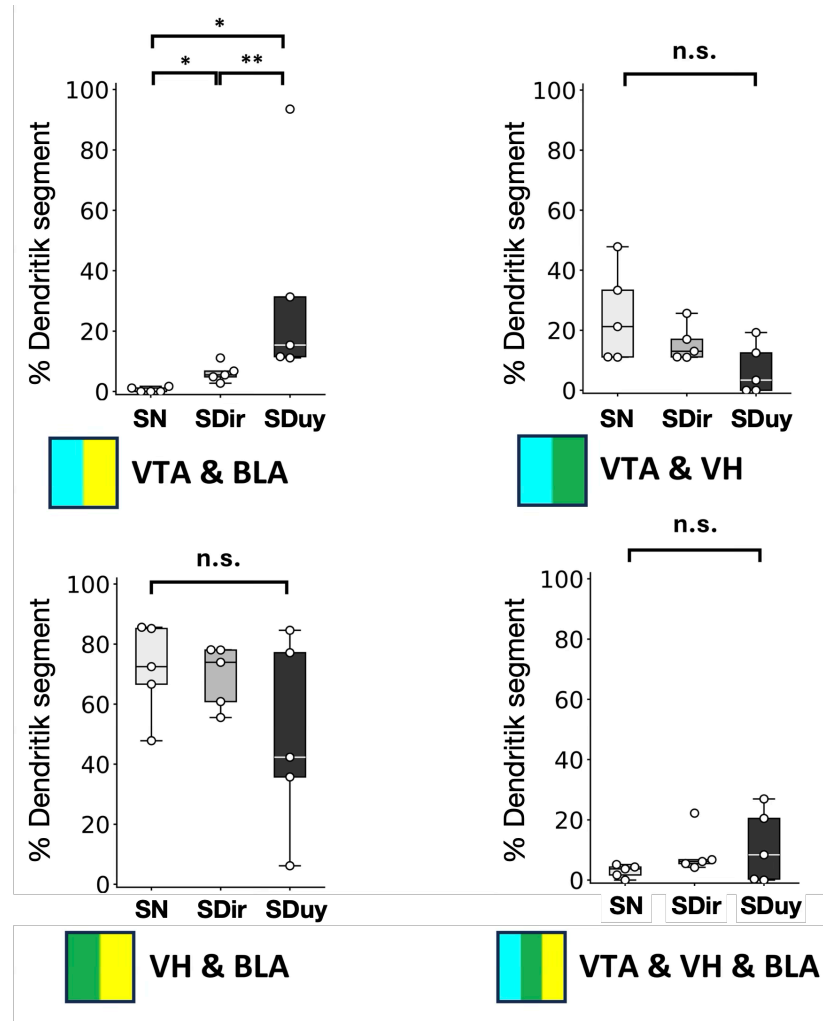
5b katmanı için,  $p = 0,61$ ,  $H = 0,96$ , Kruskal-Wallis testi. 6. katman için,  $p = 0,86$ ,  $H = 0,28$ , Kruskal-Wallis testi.

5b katmanında birden fazla beyin bölgesinden sinaptik girdi alan dendritik segmentlerin oranları gruplar arasında anlamlı fark göstermedi (Şekil 4.16.). Ancak dendritik segmentler ayrı ayrı incelendiğinde hem BLA'dan hem de VTA'dan sinaptik girdi alan dendritik segment oranlarının stresle beraber anlamlı şekilde arttığı gözlemlendi. Bu artışın SDuy farelerde SDir farelerden anlamlı şekilde daha fazla olduğu gözlemlendi (Şekil 4.17.). Diğer taraftan SDuy farelerin 6. katmanında birden fazla beyin bölgesinden girdi alan dendritik segmentlerin oranı diğer farelere göre anlamlı şekilde daha fazla bulundu (Şekil 4.16.). Dendritik segmentler ayrı ayrı incelendiğinde SDuy farelerde hem VTA hem VH'den sinaptik girdi alan dendritik segmentlerin oranının diğer farelere göre anlamlı şekilde azaldığı gözlemlendi (Şekil 4.18.). İlginç bir şekilde, her iki katmanda da hem VH hem BLA'dan sinaptik girdi alan dendritik segmentlerin oranlarında gruplar arası anlamlı bir farklılık yoktu (Şekil 4.17. ve Şekil 4.18.).

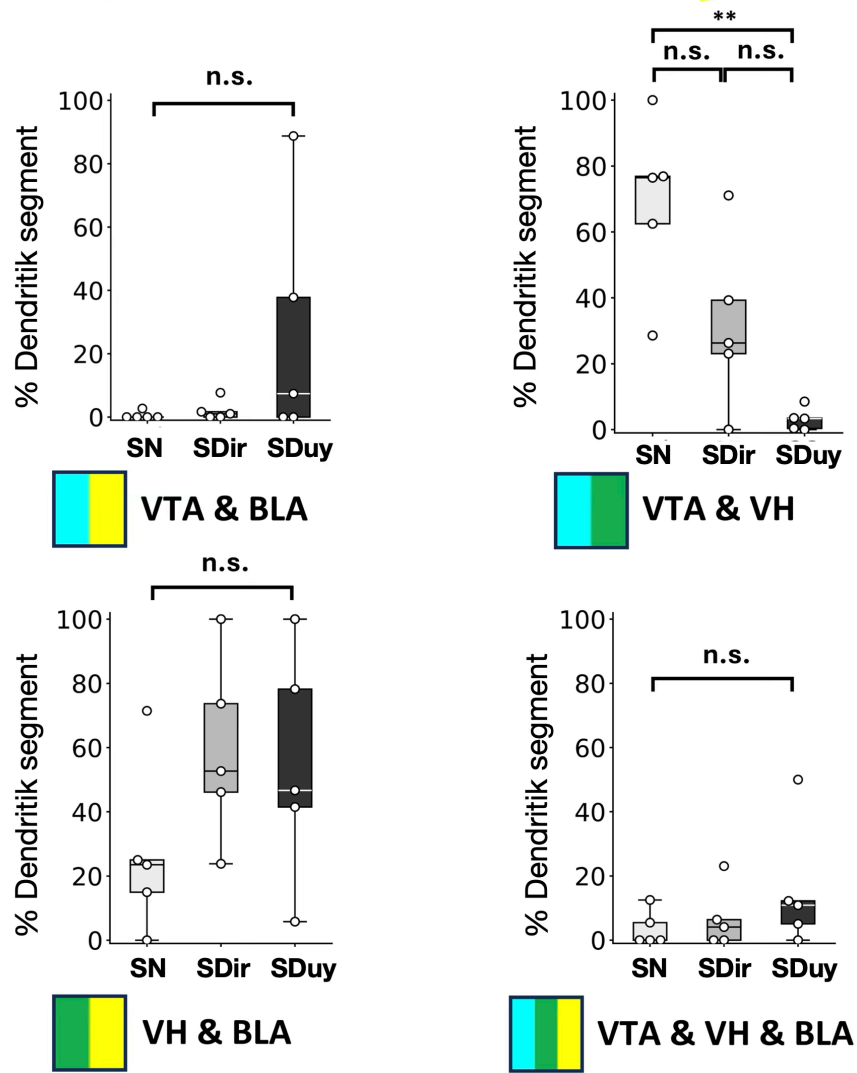


**Şekil 4.16.** 5b katmanı ve 6. katmanda birden fazla beyin bölgesinden sinaptik girdi alan dendritik segmentlerin oranlarının karşılaştırılması.

5b katmanı için,  $p = 0,22$ ,  $H = 2,96$ , Kruskal-Wallis testi. 6. katman için,  $p = 0,013$ ,  $H = 8,66$ , Kruskal-Wallis testi; SDuy-SDir için  $p = 0,015$ , SN-SDuy için  $p = 0,0079$  ve SN-SDir için  $p = 0,84$ , MWU testi. \*:  $p < 0,05$  ve \*\*:  $p < 0,01$ 'i simgelemektedir.



**Şekil 4.17.** 5b katmanında birden fazla beyin bölgesinden sinaptik girdi alan dendritik segmentlerin oranlarının her bir segment alt tipi için karşılaştırılması. VTA/BLA için,  $p = 0,0018$ ,  $H = 12,58$ , Kruskal-Wallis testi; SDuy-SDir için  $p = 0,0079$ , SN-SDuy için  $p = 0,011$  ve SN-SDir için  $p = 0,011$ , MWU testi. VTA/VH için,  $p = 0,14$ ,  $H = 3,88$ , Kruskal-Wallis testi. VH/BLA için  $p = 0,37$ ,  $H = 1,93$ , Kruskal-Wallis testi. VTA/VH/BLA için  $p = 0,19$ ,  $H = 3,31$ , Kruskal-Wallis testi. \*:  $p < 0,05$  ve \*\*:  $p < 0,01$ 'i simgelemektedir.



**Şekil 4.18.** 6. katmanda birden fazla beyin bölgesinden sinaptik girdi alan dendritik segmentlerin oranlarının her bir segment alt tipi için karşılaştırılması.

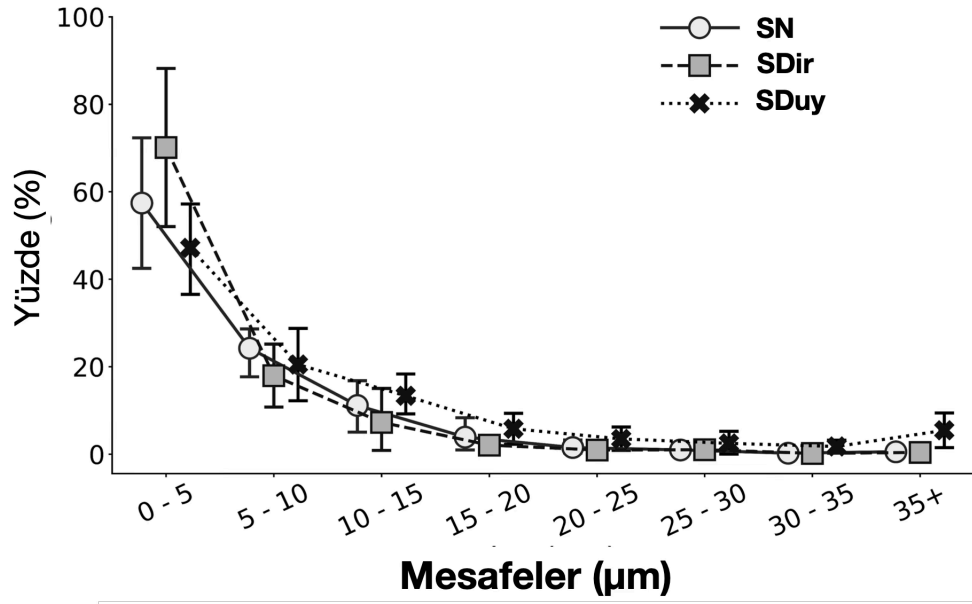
VTA/BLA için,  $p = 0,27$ ,  $H = 2,61$ , Kruskal-Wallis testi. VTA/VH için,  $p = 0,010$ ,  $H = 9,08$ , Kruskal-Wallis testi; SDuy-SDir için  $p = 0,11$ , SN-SDuy için  $p = 0,0079$  ve SN-SDir için  $p = 0,055$ , MWU testi. VH/BLA için,  $p = 0,17$ ,  $H = 3,46$ , Kruskal-Wallis testi. VTA/VH/BLA için  $p = 0,19$ ,  $H = 3,31$ , Kruskal-Wallis testi. \*\*:  $p < 0.01$ 'i simgelemektedir.

Bu bulgular, adaptif stres yanıtı sırasında stres yanıtının derin kortikal katmanlardaki sabit bir beyin bölgesinden girdi alan dendritik segmentlerin sayısını değiştirdiğini ve strese duyarlı farelerde birden fazla beyin bölgesinden sinaptik girdi alan dendritik segment sayısını değiştirerek kortikal çıktının bozulmasına sebep olabileceğini düşündürmektedir.

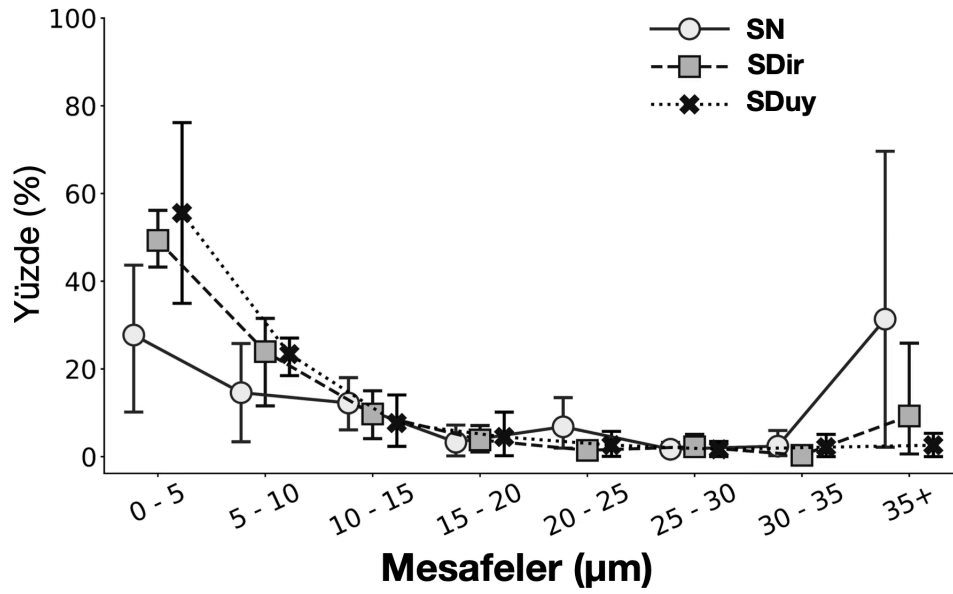
#### **4.7. mPFK derin katmanlarındaki BLA-mPFK ve VH-mPFK sinapslarının uzaysal organizasyonu SDuy farelerde bozulmuştur**

mPFK 5b ve 6. katmanında belirli bir sinaptik bağlantı organizasyonunun olup olmadığının anlaşılması için, önce her bir BLA-mPFK ve VH-mPFK sinapsı için kendisine en yakın BLA-mPFK ve VH-mPFK Öklid mesafeleri ölçüldü. BLA-mPFK sinapslarına en yakın BLA-mPFK sinaps mesafeleri ve VH-mPFK sinapslarına en yakın VH-mPFK sinaps mesafeleri gruplar arasında benzerdi (Şekil 4.19. ve Tablo 4.1). Ancak gruplar arasında VH-mPFK ve BLA-mPFK sinapslarının birbirlerine olan uzaklıklarında farklılıklar saptandı. SN ve SDir farelerde VH-mPFK sinapslarına en yakın BLA-mPFK sinaps mesafeleri, farklı mesafe aralıklarında eşit şekilde dağılırken; BLA-mPFK sinapslarına en yakın VH-mPFK sinaps mesafelerinin oranı kısa mesafe aralığında (0-5  $\mu\text{m}$ ) daha fazla idi. SDuy grupta ise, VH-mPFK sinapsları ile en yakın BLA-mPFK sinapsları arasındaki mesafede azalma ve BLA-mPFK sinapsları ile en yakın VH-mPFK sinapsları arasındaki mesafede artış gözlemlendi (Şekil 4.20. ve Tablo 4.2).

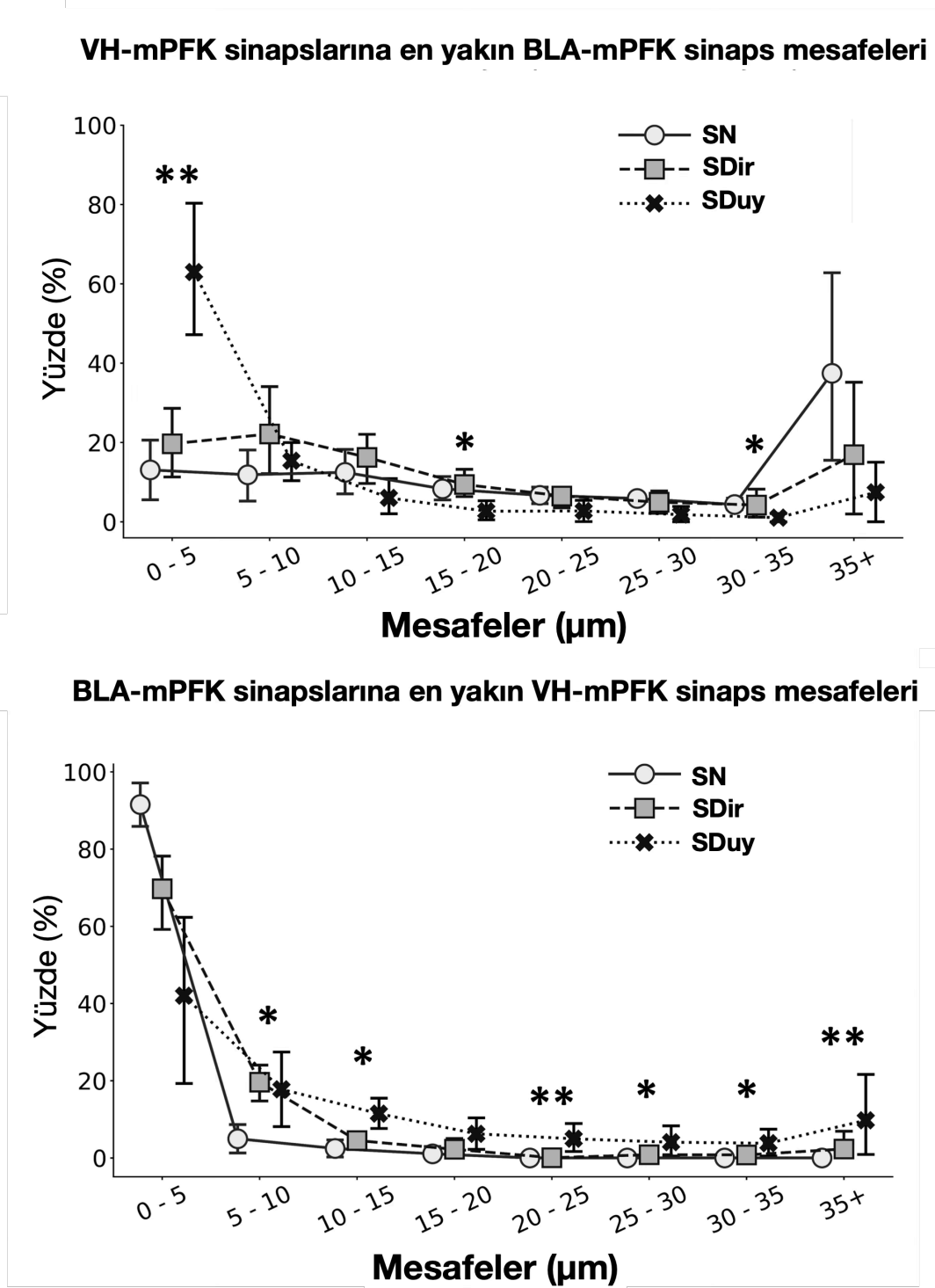
### VH-mPFK sinapslarına en yakın VH-mPFK sinaps mesafeleri



### BLA-mPFK sinapslarına en yakın BLA-mPFK sinaps mesafeleri



**Şekil 4.19.** 5b ve 6. katmanda VH-mPFK sinapslarına en yakın VH-mPFK sinapslarının ve BLA-mPFK sinapslarına en yakın BLA-mPFK sinapslarının tüm gruplar için histogram dağılımları.



**Şekil 4.20.** 5b ve 6. katmanda VH-mPFK sinaplarına en yakın BLA-mPFK sinapslarının ve BLA-mPFK sinaplarına en yakın VH-mPFK sinapslarının tüm gruplar için histogram dağılımları. \*:  $p < 0,05$  ve \*\*:  $p < 0,01$ 'i simgelemektedir.

**Tablo 4.1.** VH-mPFK sinapsına en yakın VH-mPFK sinapsı ve BLA-mPFK sinapsına en yakın BLA-mPFK sinaps mesafelerinin gruplar arasındaki istatistiksel karşılaştırma sonuçları (Bkz. Şekil 4.19.).

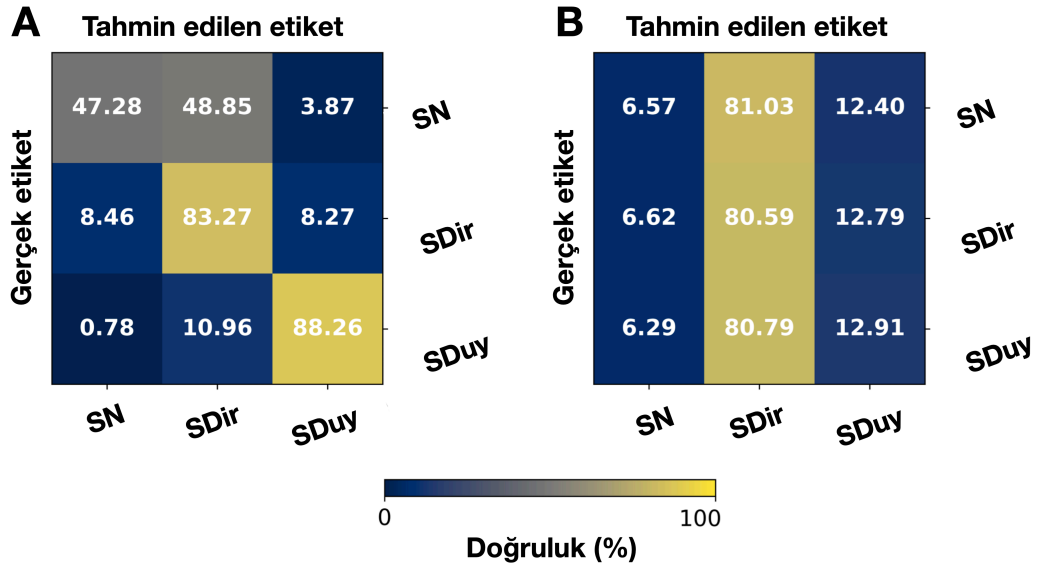
Katman	Aralık (µm)	KW	MWU testi			
			SN vs STR	SDuy vs SDir	SDuy vs SN	SDir vs SN
<b>VH-mPFK sinapsına en yakın VH-mPFK sinapsı</b>						
2/3. Katman	0-5	0.67	1.00			Yapılmadı
	5-10	0.98	0.95			Yapılmadı
	10-15	0.24	0.32			Yapılmadı
	15-20	<b>0.056</b>	<b>0.022</b>	<b>0.51</b>	<b>0.059</b>	<b>0.059</b>
	20-25	0.55	0.67			Yapılmadı
	25-30	0.73	0.84			Yapılmadı
	30-35	0.55	0.31			Yapılmadı
	35+	0.35	0.66			Yapılmadı
	5b ve 6. Katman	0-5	0.29	1.00		
5-10		0.45	0.25			Yapılmadı
10-15		0.53	0.76			Yapılmadı
15-20		0.26	0.95			Yapılmadı
20-25		0.29	0.80			Yapılmadı
25-30		0.80	0.95			Yapılmadı
30-35		0.24	0.62			Yapılmadı
35+		0.15	0.75			Yapılmadı
<b>BLA-mPFK sinapsına en yakın BLA-mPFK sinapsı</b>						
2/3. Katman	0-5	0.20	0.08			Yapılmadı
	5-10	0.69	0.53			Yapılmadı
	10-15	0.72	0.80			Yapılmadı
	15-20	0.37	0.20			Yapılmadı
	20-25	0.48	0.25			Yapılmadı
	25-30	0.39	0.24			Yapılmadı
	30-35	0.31	0.22			Yapılmadı
	35+	0.86	0.64			Yapılmadı
	5b ve 6. Katman	0-5	0.22	0.09		
5-10		0.28	0.24			Yapılmadı
10-15		0.67	0.46			Yapılmadı
15-20		0.96	0.85			Yapılmadı
20-25		0.40	0.21			Yapılmadı
25-30		0.81	0.71			Yapılmadı
30-35		0.53	0.38			Yapılmadı
35+		0.18	0.12			Yapılmadı



**Tablo 4.2.** VH-mPFK sinapsına en yakın BLA-mPFK sinapsı ve BLA-mPFK sinapsına en yakın VH-mPFK sinaps mesafelerinin gruplar arasındaki istatistiksel karşılaştırma sonuçları (Bkz. Şekil 4.20.).

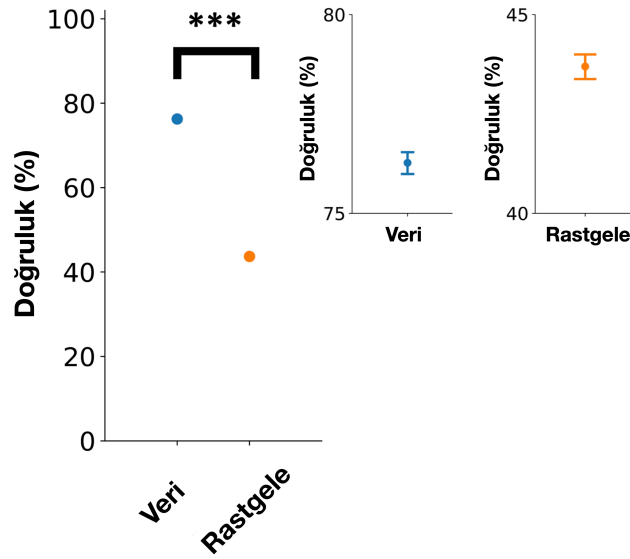
Katman	Aralık ( $\mu\text{m}$ )	KW	MWU testi			
			SN vs STR	SDuy vs SDir	SDuy vs SN	SDir vs SN
<b>VH-mPFK sinapsına en yakın BLA-mPFK sinapsı</b>						
2/3. Katman	0-5	0.06	0.055	0.42	0.54	0.0079
	5-10	0.94	0.80		Yapılmadı	
	10-15	0.27	0.16		Yapılmadı	
	15-20	0.12	0.13		Yapılmadı	
	20-25	<b>0.022</b>	<b>0.0086</b>	<b>0.42</b>	<b>0.07</b>	<b>0.025</b>
	25-30	<b>0.031</b>	<b>0.013</b>	<b>0.42</b>	<b>0.11</b>	<b>0.025</b>
	30-35	<b>0.022</b>	<b>0.0086</b>	<b>0.42</b>	<b>0.07</b>	<b>0.025</b>
	35+	0.13	0.06		Yapılmadı	
5b ve 6. Katman	0-5	<b>0.009</b>	<b>0.027</b>	<b>0.015</b>	<b>0.0079</b>	<b>0.30</b>
	5-10	0.35	0.30		Yapılmadı	
	10-15	0.10	1.00		Yapılmadı	
	15-20	<b>0.034</b>	<b>0.37</b>	<b>0.031</b>	<b>0.031</b>	<b>0.69</b>
	20-25	0.14	0.50		Yapılmadı	
	25-30	0.06	0.07	0.13	0.034	0.42
	30-35	<b>0.036</b>	<b>0.055</b>	<b>0.16</b>	<b>0.011</b>	<b>0.54</b>
	35+	0.06	0.041	0.33	0.034	0.22
<b>BLA-mPFK sinapsına en yakın VH-mPFK sinapsı</b>						
2/3. Katman	0-5	0.19	0.11		Yapılmadı	
	5-10	0.92	0.75		Yapılmadı	
	10-15	0.43	0.21		Yapılmadı	
	15-20	0.24	0.25		Yapılmadı	
	20-25	0.88	0.71		Yapılmadı	
	25-30	0.73	0.49		Yapılmadı	
	30-35	0.41	0.94		Yapılmadı	
	35+	0.53	0.53		Yapılmadı	
5b ve 6. Katman	0-5	<b>0.004</b>	<b>0.00066</b>	<b>0.055</b>	<b>0.0079</b>	<b>0.0079</b>
	5-10	<b>0.045</b>	<b>0.012</b>	<b>0.84</b>	<b>0.09</b>	<b>0.015</b>
	10-15	<b>0.016</b>	<b>0.031</b>	<b>0.031</b>	<b>0.021</b>	<b>0.20</b>
	15-20	0.07	0.06	0.30	0.034	0.33
	20-25	<b>0.004</b>	<b>0.045</b>	<b>0.017</b>	<b>0.0074</b>	<b>0.42</b>
	25-30	<b>0.046</b>	<b>0.045</b>	<b>0.23</b>	<b>0.025</b>	<b>0.17</b>
	30-35	<b>0.031</b>	<b>0.08</b>	<b>0.11</b>	<b>0.025</b>	<b>0.42</b>
	35+	0.10	0.13		Yapılmadı	

Bu sonuçlar gruplar arasında gözlenen BLA ve VH kaynaklı mPFK aksonal uzanımların rastgele dağılmadığını, yapılandırılmış bir düzeni olduğunu düşündürmektedir. Bu varsayımı test etmek için, 5b ve 6. katmandaki bütün BLA-mPFK ve VH-mPFK sinapslarına Öklid mesafesi en yakın ilk 4 BLA-mPFK ve ilk 4 VH-mPFK sinapslarını içeren 42.317 x 8 boyutunda matris oluşturuldu. Bu matriste ilk dört kolon sinapslarına en yakın VH-mPFK sinapslarına; sonraki dört kolon en yakın BLA-mPFK sinapslarını içerecek şekilde düzenlendi. Ardından bu matris rastgele orman sınıflamasından (*Random Forest Classification, ROS*) geçirildi. ROS SDir sinapslarının %83, SDir sinapslarının %88 doğrulukla tespitini yapabildi ancak SN gruptaki sinapsların tespitini %47 doğrulukla yapabildi (Şekil 4.21.). ROS, sinapsların sahip olduğu stres duyarlılık etiketi rastgele dağıtılarak tekrarlandığında hem doğru sınıflama oranının hem de F1 skorunun anlamlı şekilde azaldığı gözlemlendi (Şekil 4.22. ve Şekil 4.23.).

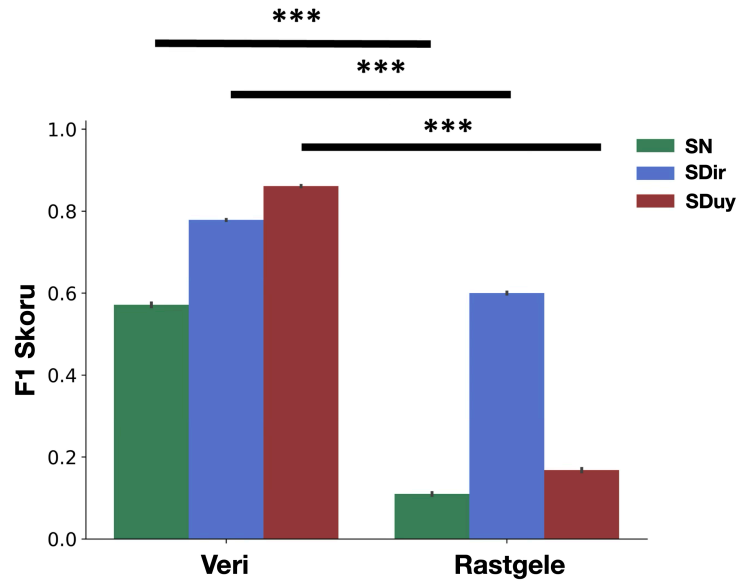


**Şekil 4.21.** 5b ve 6. katmandaki BLA-mPFK ve VH-mPFK sinapslarının ilk dört BLA-mPFK ve ilk dört VH mPFK sinapslarına en yakın Öklid mesafelerine göre rastgele orman sınıflamasıyla stres duyarlılık sınıflaması.

**A)** Veri etiketlerinin orijinal haliyle incelendiği durum; **B)** Veri etiketlerinin rastgele dağıldığı durum.

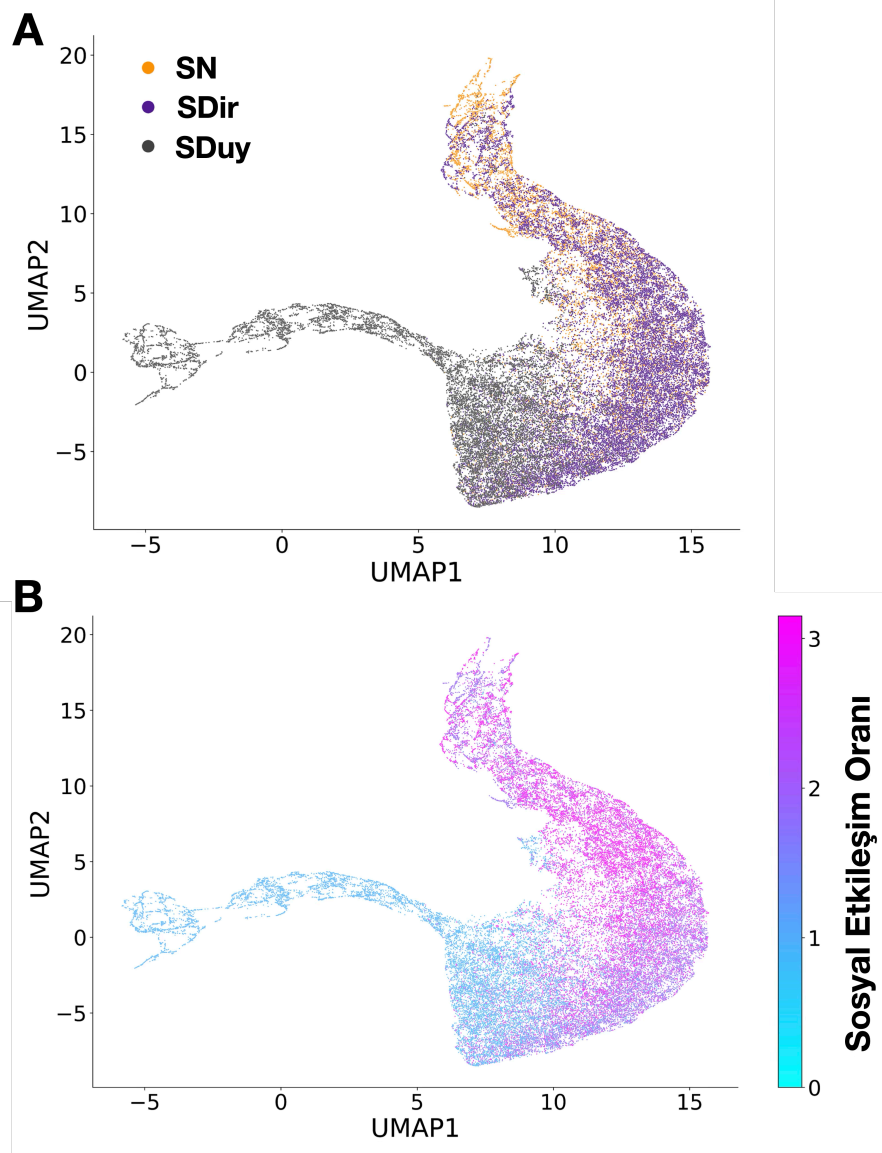


**Şekil 4.22.** ROS'nin doğrulama yüzdesinin elde edilen ve davranış fenotip etiketleri rastgele dağılan veriler için karşılaştırılması.  
Veri-rastgele karşılaştırması,  $p = 5,16 \times 10^{-5}$ , bağımsız örneklerde t-testi.



**Şekil 4.23.** ROS F1 skorlarının elde edilen ve davranış fenotip etiketleri rastgele dağılan veriler için karşılaştırılması.  
SN fare için,  $p = 1,83 \times 10^{-24}$ ; SDir fare için,  $p = 2,34 \times 10^{-23}$  ve SDuy fare için,  $p = 2,12 \times 10^{-30}$ , bağımsız örneklerde t-testi.

Son olarak bu matris doğrusal olmayan boyut indirgeme yöntemi, UMAP (*uniform manifold approximation and projection*) kullanılarak (McInnes ve ark, 2018) 2 boyutta görüntülediğinde SDir ve SDuy farelerin 2 boyutlu UMAP düzleminde farklı alanları işgal ettiği gözlemlendi (Şekil 4.24.).



**Şekil 4.24.** 5b ve 6. katmandaki BLA-mPFK ve VH-mPFK sinapslarının ilk dört BLA-mPFK ve ilk dört VH mPFK sinapslarına en yakın Öklid mesafelerinin UMAP projeksiyonu.

**A)** strese duyarlılık fenotipi ve **B)** sosyal etkileşim oranına göre düzenlenmiştir.

## 5. TARTIŞMA

Stres yanıtı, hayvanların davranış repertuvarının temel bir bileşenidir, bununla beraber halen sağlık ve hastalık durumlarında mekanistik, devresel düzeyde farklılıklar net olarak ortaya konamamıştır. Bunun bir kısmı, davranışsal yanıtları yöneten dağıtılmış devre yapısının karmaşıklığından kaynaklanmaktadır; bu yapının içinde birden çok beyin bölgesi arasında çok sayıda sinaptik bağlantı bulunmaktadır. Bu çalışmada, çok renkli eGRASP proteinleri, multispektral volümetrik görüntüleme, dendrit ve sinaps tanımlaması için hesaplama araçları ve elde edilen büyük ölçekli veri setini analiz etmek için istatistiksel yöntemler kullanılarak, bu devre yapısının fare mPFC'yi üzerinde aydınlatılması ve strese duyarlı ve dirençli fare beyinleri arasındaki farkların ortaya konması amaçlandı. Bu çalışma, VH-mPFC, VTA-mPFC ve BLA-mPFC sinaptik bağlantılarını ayrı ayrı tüm kortikal katmanlar boyunca haritalayan ve bunları SN, SDir ve SDuy fareler arasında karşılaştıran ilk çalışmadır. Bu çok yönlü yaklaşım, stres yanıtı ve stres duyarlılığında derin ve yüzeysel mPFC katmanlarındaki sinaptik girdilerin farklı bağlanma desenlerini ortaya koymuştur.

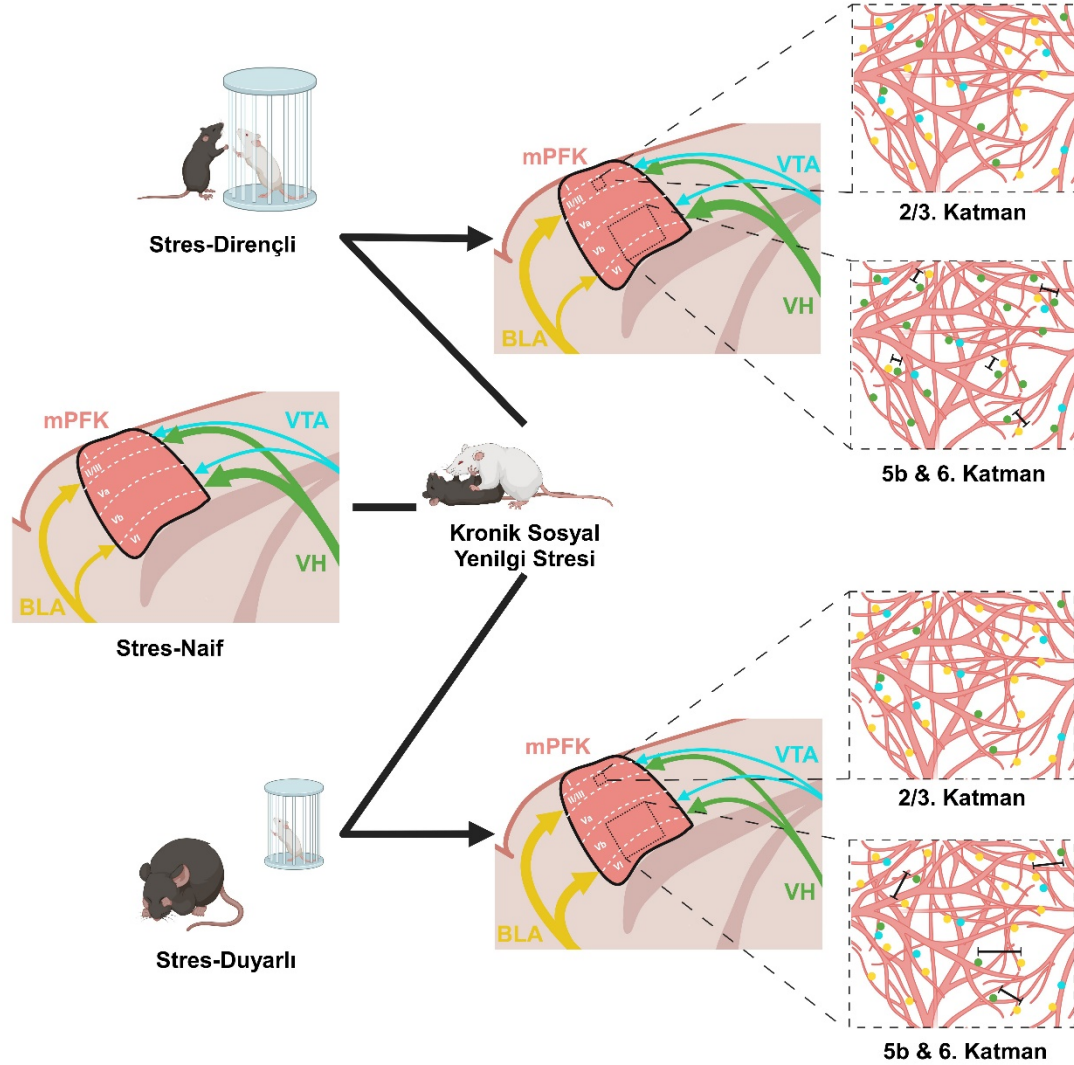
Önceki çalışmalarda, BLA projeksiyonlarının ağırlıklı olarak mPFC'nin 2. katmanında sonlandığı gösterilmiştir (24, 25, 146). Diğer yandan VH projeksiyonlarının genellikle mPFC'nin 5. katmanına sınırlı olduğu gösterilmiştir (25, 26). Biz de literatürdeki bu bulgularla uyumlu olarak, SN farelerinin mPFC katmanlarında sonlanan uzun menzilli sinaptik girdilerin benzer bir dağılımı olduğunu gözlemledik. KSYS, farelerin stres duyarlılığından bağımsız şekilde mPFC yüzeysel katmanlarındaki BLA'dan gelen sinaptik girdilerin oranlarını  $[BLA/(BLA+VH+VTA)]$  ve VH'den gelen sinaptik girdilerin oranlarını  $[VH/(BLA+VH+VTA)]$  değiştirdi. Diğer yandan, uzun menzilli sinaptik girdilerin mPFC'nin derin katmanlarındaki dağılımı, yalnızca SDuy grubunda bozuldu: BLA-mPFC sinapsları, 5b ve 6. katmanlarda VH-mPFC sinapslarına üstünlük sağladı; bu da uzun menzilli sinaptik girdilerin mPFC'nin derin katmanlarında yaptıkları sinaptik bağlantıların organizasyonundaki farklılıkların stres duyarlılığının altında yatabileceğini düşündürmektedir. Buna karşılık, VTA'dan gelen sinaptik girdilerin  $[VTA/(BLA+VH+VTA)]$  oranı tüm kortikal katmanlar ve gruplar arasında benzerdi.

KSYS'ye maruz kalan tüm farelerde, mPFK 2/3. katmanında BLA-mPFK sinapslarının bolluğunda artış ve VH-mPFK sinapslarının bolluğunda azalma gözlemlendi. Bu durum, bu katmanda gözlenen değişikliklerin kronik strese maruz kalma ile ilişkili olduğunu göstermektedir. Önceki çalışmalarda, kronik strese maruz bırakılan kemirgenlerde mPFK'nin 2/3. ve 5. katmanlarında dendrit uzunluklarında ve dendritik çıkıntı sayılarında azalmalar bildirilmiştir (2-6). Sonuçlarımız, bu bulgularla birlikte değerlendirildiğinde, kronik stresin VH-mPFK sinapslarının atrofiye uğramasına yol açtığını ve bu nedenle VH-mPFK sinapslarında azalma olduğunu; ancak BLA-mPFK sinapslarının kronik stresten etkilenmediğini düşündürmektedir. Bu öneriyi destekleyici şekilde, Shansky ve arkadaşları (8), kronik stres yanıtında entorinal kortekse projeksiyon yapan piramidal nöronların dendritik atrofi gösterdiğini, BLA'ya projeksiyon yapanların ise stres kaynaklı dendritik yeniden şekillenmeye karşı dirençli olduğunu göstermiştir. BLA ve mPFK arasında karşılıklı bir bağlantı olduğu düşünüldüğünde (yani BLA girdileri alan nöronların tercihen BLA'ya projeksiyon yapan mPFK nöronlarını inerve etmesi), BLA'dan girdi alan mPFK nöronlarının da atrofiye karşı dirençli olduğu kestirilebilir (24, 25, 146, 147).

mPFK, BLA ve VH'yi birbirine bağlayan beyin ağları, korkuyla ilişkili ve sosyal davranışların düzenlenmesinde rol oynamaktadır (15, 18, 148-151). Her üç bölge de anksiyetenin ve sosyal davranışların kontrolünde rol oynar, aralarındaki etkileşim karmaşıktır; korkuyu da sosyal davranışı da hem artırabilir hem de bastırabilirler (18, 25, 27, 147, 152, 153). mPFK'ye BLA ve VH'den köken alan sinaptik girdilerin, korkuyla ilişkili ve sosyal davranışlar üzerinde zıt etkilere sahip olduğu bildirilmiştir (15, 18, 61). Bunun yanı sıra, her bir bölge içinde farklı bağlantılara sahip, "güvenli" ve "tehditle ilişkili" ipuçlarını kodlayan ayrı nöron popülasyonları bulunduğunu gösteren çalışmalar mevcuttur. Bu düzenleme, organizmanın çeşitli uyaranlara optimal tepki vermesini sağlar (27, 28, 147). Farklı nöronal devrelerin seçici aktivasyonu; duyuşsal, bağlamsal ve duygusal bilgileri entegre ederek alternatif davranış yanıtları arasında geçiş yapılmasını sağlayabilir, korkuyla ilişkili ve sosyal yanıtların ifade edilmesini ve bastırılmasını hassas bir şekilde ayarlayabilir (22, 27, 153).

Fareler sosyal hayvanlardır, sosyal hedeflerle (yabancı bir fare) sosyal olmayan hedeflere (nesnelere) kıyasla daha fazla zaman geçirmeyi tercih ederler (154). KSYS sonrasında, SDir ve SDuy fareler daha önce karşılaşmadıkları bir CD1 fare ile yeni bir bağlamda karşılaştıklarında zıt davranışsal yanıtlar gösterirler: SDuy fareleri sosyal kaçınma sergilerken, SDir fareleri sosyal yaklaşma gösterirler. Bu, SDir farelerinin tehditle ilişkili uyaran veya bağlamları yeni uyaran veya bağlamlardan ayırabildiğini, SDuy farelerinin de tehdidi karşılaştıkları tüm yeni uyaran veya bağlamlara genelleme eğiliminde olduğunu düşündürmektedir. Korkunun genelleştirilmesi (*overgeneralization of fear*) anksiyete bozukluklarının karakteristik bir özelliği olup, travma sonrası stres bozukluğu, panik bozukluğu ve yaygın kaygı bozukluğunun altında yatan mekanizmalardan biri olarak düşünülmektedir (155-158). Bu çalışmada bildirilen, SDuy farelerinde mPFK piramidal nöronlarının derin katmanlarına BLA'dan gelen uzun menzilli sinaptik girdilerin yoğunluğundaki göreceli artış ve VH'dan gelen uzun menzilli sinaptik girdilerin yoğunluğundaki göreceli azalma, tehditle ilişkili ve tehditle ilişkilendirilmeyen uyaranları birbirinden ayırmasını, dolayısıyla mPFK'nin adaptif bir yanıt üretme kapasitesini bozabilir. Daha önce bu varsayımı destekler nitelikteki bir çalışma, tehdit ipuçlarının güvenli ipuçlarından ayırt edilmesinin, mPFK ve BLA aktiviteleri arasındaki teta frekans aralığındaki eşleşme (senkronizasyon) ile ilişkili olduğunu göstermiştir (153). mPFK aktivitesinin teta frekans aralığında BLA aktivitesini sürüklemesi (*entrainment*) sadece koşullu işitsel tonları (*conditioned auditory tones*) nötr (koşullandırılmamış) işitsel tonlardan ayırt edebilen farelerde bildirilmiştir. Bu farelerde, nötr tonlara kıyasla koşullandırılmış işitsel tonlara karşı daha fazla donma yanıtı gözlenmiştir (153). Bunun yanı sıra, anksiyojenik yanıt oluşturabilen yeni ortamları keşfederken VH aktivitesi teta frekans aralığında mPFK aktivitesiyle senkronize olur ve güvenli ve tehditle ilişkili ipuçlarının ayırt edilmesini kolaylaştırır (19, 20, 22). VH'nin ayrıca sosyal hafıza oluşumunda ve depolanmasında da rol oynadığı düşünülmektedir (150). VH nöronları, yabancı bir fareye kıyasla tanıdık bir fare karşısında daha güçlü bir şekilde aktive olur, VH nöronlarının optogenetik inhibisyonu tanıdık ve yabancı farelerin ayırt edilmesini engeller (150). Bu bulgulara paralel olarak, bu tez çalışmasının sonuçları, BLA-mPFK ve VH-mPFK sinapsları arasındaki dengenin, sosyal işaretleri tehditle ilişkili ipuçlarıyla ilişkilendirmede ve en uygun yanıtı verebilmek için güvenli ipuçlarını

tehditle ilişkili ipuçlarından ayırt etmede önemli olduğuna işaret etmektedir (Şekil 5.1.).



**Şekil 5.1.** Çalışmanın özet şekli.

Bu tez çalışmasında mPFC'nin eferent bağlantıları araştırılmadığı için, SDuy farelerde mPFC'nin derin katmanlarındaki uzun menzilli girdilerdeki düzensizliğin hangi beyin bölgesini/bölgelerini etkilediği bilinmemektedir. Ancak, şimdiye kadar yapılmış çalışmalar, mPFC'nin akübens çekirdeğe (NAk) projeksiyonlarının sosyal yaklaşma davranışında rol oynadığını düşündürmektedir (159). SDuy farelerinde uzun menzilli sinaptik girdi organizasyonunun sadece mPFC'nin derin katmanlarında bozulduğu ve bu katmandaki piramidal nöronların subkortikal bölgelere eferent verdiği göz önüne alındığında, BLA ve VH sinaptik girdilerindeki dengesizliğin



mPFC-NAK devresindeki aktiviteyi bozması olasıdır. Bu konunun gelecek çalışmalarda ele alınması gereklidir.

Farklı beyin bölgelerinden mPFC'ye gelen eksitator sinaptik girdilerin farklı stres fenotiplerine nasıl katkı sağladığı önemli bir sorudur. Bu girdiler, belirli nöronal toplulukları (*ensemble*) seçici olarak hedef alabilir ve bu toplulukların her birinin korkuyla ilişkili davranışlarda farklı rolleri olabilir (160-162). Bu öneriyi destekleyecek şekilde, bulgularımız SN ve SD farelerde aynı beyin bölgesinden gelen girdilerin aynı dendritik segment üzerinde sinaps oluşturduğunu gösterdi. SD farelerde ise, farklı beyin bölgelerinden gelen sinaptik girdilerin aynı dendritik segment üzerinde yaptığı sinaps sayısında artış gözlemlendi. Ayrıca, verilerimiz normal fizyolojik koşullarda, BLA'dan gelen girdilerin genellikle VH girdilerine yakın noktalarda sonlandığını bununla birlikte VH-mPFC sinapslarının bir bölümünün BLA-mPFC sinapslarından daha uzakta konumlanmasının gerekli olduğunu gösterdi. SD farelerde, bu innervasyon paterni bozulmuş ve neredeyse tüm VH girdilerinin BLA girdilerine yakın noktalarda sonlandığı gözlenmiştir. Bu bulgulardan hareketle, BLA girdilerinin VH girdilerine yakınlığının, bu nöronal topluluk içinde devam eden bilgi işleme sürecine tehditle ilişkili ipuçlarını entegre etmede rol oynayabileceği, BLA girdilerinden uzakta olan VH girdilerinin ise tehditle ilişkili olmayan ipuçlarını işlemede rol alabileceği öne sürülebilir. Bu farklı girdilerin mPFC'ye projekte olan ve her biri korkuyla ilişkili davranışlarda karşıt rollerde görev alan farklı VH nöron popülasyonlarından mı kaynaklandığı sorusu gelecek çalışmalarda ele alınması gereken bir konudur (28). İlginç bir şekilde, derin mPFC katmanlarında bildirilen VH ve BLA sinapslarındaki organizasyon bozukluğu tek dendrit düzeyinde yapılan analizlerde görülmedi, hem VH hem BLA'dan sinaptik girdi alan dendritik segment yüzdeleri gruplar arasında farklılık göstermedi.

Önceki çalışmalarda VTA'nın hem stres yanıtı hem de sosyal davranışları düzenlemedeki rolün bildirilmiş olmakla beraber, bu tez çalışmasında VTA'nın toplam sinaptik girdilere oranında ne kronik stresle ne de stres duyarlılığı ile herhangi bir değişiklik gözlemlenmedi. Bu uyumsuzluk, önceki çoğu çalışmanın dopaminerjik nöronlara odaklanmasından; ancak bu tez çalışmasında glutamaterjik VTA-mPFC projeksiyonlarının incelenmesinden kaynaklanmış olabilir (12, 13, 67). Yine de,

birden fazla beyin bölgesinden sinaptik girdi alan dendritik segmentlerde, VTA'nın yer aldığı ikililerin stres duyarlılığı ile ilişkili olduğu gözlemlendi. Eğer VTA-mPFC projeksiyonlarının dopaminerjik bir özelliği varsa bu durum BLA ve VH sinapslarının birlikte olduğu dopamin reseptör alt tipi ile ilişkili olabilir (63, 64, 103, 104, 163). D1R veya D2R aktivasyonu ile aynı dendrit dalındaki BLA veya VH sinapsının dendritik aksiyon potansiyelinin oluşturma ihtimali değişikliğiyle stres duyarlı veya dirençli fenotip gözlemlenebilir (23, 164). Bu durumun anlaşılması, D1R ve D2R pozitif piramidal nöronlarındaki BLA ve VH sinapslarının dağılımının incelenmesiyle mümkün görünmektedir.

Başka bir olasılık VTA projeksiyonlarının tercihen ara nöronlarla sinaps oluşturması olabilir (65). Ancak bu tez çalışmasında ara nöronlar üzerindeki bağlantılar incelenmemiştir. Stresle ilişkili ruhsal bozukluklarda değişmiş eksitasyon:inhibisyon (E:I) dengesi hakkında çeşitli çalışmalar göz önüne alındığında, mPFC ara nöronlarına gelen uzun menzilli projeksiyonlar stres yanıtının ince ayarına katkıda bulunabilir (165-169). Bu, VTA-mPFC sinaptik girdilerinin stres tepkisinin düzenlenmesinde de rol oynayabileceğini düşündürmektedir. VTA'nın özellikle mPFC ara nöronları üzerinde sonlanan glutamaterjik projeksiyonlarının stres yanıtındaki rolünün gelecek çalışmalarda daha detaylı incelenmesi gereklidir.

Bu çalışmanın bir kısıtlılığı sadece erkek farelerde yapılmış olmasıdır, bu da bulgularımızın dişilere genelleştirilebilmesini sınırlamaktadır. KSYS dişi farelerde uygulanması zor olduğu için, genellikle erkek farelerde stres fenotipinin ayrıştırılmasında kullanılmaktadır. Ne erkek ne de dişi fareler yabancı dişilere saldırmazlar, bu nedenle, erkek farelerin aksine, dişiler doğal yollarla bu tür sosyal saldırılara maruz kalmazlar. Dişilerde sosyal yenilgiyi indüklemek amacıyla, CD1 erkek fareye ait idrarın dişi farelerin vajinal açıklıklarına uygulanmasıyla bir KSYS modeli geliştirilmiş olmasına rağmen, modelin etolojik geçerliliğinin erkek farelerde daha yüksek olması nedeniyle çalışmanın sadece erkek farelerde yapılması tercih edilmiştir (170).

İkinci sınırlılık, çalışmanın piramidal mPFC nöronlarına olan uzun menzilli projeksiyonlara odaklanması ve mPFC ara nöronlarında sonlanan projeksiyonların incelenmemiş olmasıdır. Bu karar, özellikle birden fazla floroforun aynı anda konfokal

çok renkli görüntülenme için kantitatif sınırlar nedeniyle alınmıştır. Gelecekteki çalışmalarda stres yanıtı ve strese direncin temelinde yatan nöronal devrelerinin daha kapsamlı bir resmini elde etmek için mPFK ara nöronlarına olan uzun menzilli projeksiyonlar da araştırılmalıdır.

Sonuç olarak bu çalışma, mPFK'ye gelen uzun menzilli projeksiyonların oluşturduğu sinapslara odaklanarak hem fizyolojik hem de patolojik durumlarda kronik stresin beyin devreleri üzerindeki etkilerini daha iyi anlamamıza katkı sağlamıştır. Strese maruz bırakılan gruplarda stres fenotipine (dirençli vs. duyarlı) bağlı şekilde kortikal katmana ve afferente spesifik değişiklikler gösterildi. Bulgularımızdan hareketle planlanacak yeni çalışmalarla strese duyarlı bireylerde stresli veya travmatik bir olayın ardından önleyici veya terapötik stratejiler geliştirilebilir.

## 6. SONUÇLAR VE ÖNERİLER

- Enjeksiyondan başlayarak görüntü işlemeye kadar fare beyinlerinde mPFC tüm katmanlarında ışık mikroskopisi kullanarak üç beyin bölgesinden kaynaklı sinapsların dendrit üzerinde haritalaması için yöntem oluşturulmuştur.
- Kronik stres maruziyetiyle mPFC piramidal nöron sinapslarında beyin bölgesi ve kortikal katman spesifik değişikliklerin strese duyarlılıkta rol oynadığı gösterilmiştir.
- mPFC 2/3. katmanındaki BLA-mPFC sinapslarının görelî oranın kronik strese maruz kalma ile arttığı; mPFC 5. katman derini ve 6. katmanındaki BLA-mPFC sinapslarının görelî oranının sadece strese duyarlı farelerde arttığı gözlenmiştir.
- mPFC 2/3. katmanındaki VH-mPFC sinapslarının görelî oranın kronik strese maruz kalma ile azaldığı; mPFC 5. katman derini ve 6. katmanındaki VH-mPFC sinapslarının görelî oranının sadece strese duyarlı farelerde azaldığı gözlenmiştir.
- mPFC'nin bütün kortikal katmanlarında kronik strese maruz kalma ile VTA-mPFC sinapslarının görelî oranıyla ilişkili bir değişiklik gözlenmemiştir.
- mPFC 6. katmanında birden fazla beyin bölgesinden sinaptik girdi alan dentritik segmentlerin görelî oranının sadece strese duyarlı farelerde arttığı gözlenmiştir.
- mPFC 5. katman derininde aynı anda VH ve VTA'dan sinaptik girdi alan dentritik segmentlerin görelî oranının sadece strese duyarlı farelerde azaldığı; 6. katmanda aynı anda BLA ve VTA'dan sinaptik girdi alan dentritik segmentlerin görelî oranının sadece strese duyarlı farelerde arttığı gözlenmiştir.
- mPFC 5. katman derini ve 6. katmanda bulunan VH-mPFC ve BLA-mPFC sinapslarının uzaysal ilişkilerinin sadece strese duyarlı farelerde bozulduğu gözlenmiştir.
- Ara nöronların piramidal nöronların aktivitesini düzenlediğinden ve diğeri beyin bölgelerinden sinaptik girdiğî almasından dolayı sonraki çalışmalarda bu

haritalamanın sadece piramidal nöronlara değil, ara nöronlara da genişletilmesi gerekmektedir.

- Anti-depresan tedavi stratejilerini ve/veya optogenetik/kemogenetik aktivasyon/inhibisyon stratejilerinin bu çalışmada gözlemlenen sinaptik dağılım değişikliklerini geri döndürüp döndüremeyeceği incelenmelidir.
- Bu çalışmada gözlemlenen değişikliklerinin piramidal nöron çıktı nöronları (IT ve PT nöronları) ve beyindeki çıktı bölgeleriyle ilişkisi incelenmelidir.
- Bu çalışmada sadece BLA'daki görece aktivite artışının değil, VH'daki görece aktivite kaybının da strese duyarlılıkta rol oynadığı gözlenmiştir. Sonraki yapılacak *in vivo* optogenetik deneylerinde bu durumun göz önünde tutularak birden fazla devreye aynı anda müdahale seçeneklerinin bulunduğu deneylerin planlanması gereklidir.

## 7. KAYNAKLAR

1. Cathomas F, Murrrough JW, Nestler EJ, Han MH, Russo SJ. Neurobiology of Resilience: Interface Between Mind and Body. *Biol Psychiatry*. 2019;86(6):410-20.
2. Cook SC, Wellman CL. Chronic stress alters dendritic morphology in rat medial prefrontal cortex. *J Neurobiol*. 2004;60(2):236-48.
3. Liston C, Miller MM, Goldwater DS, Radley JJ, Rocher AB, Hof PR, et al. Stress-induced alterations in prefrontal cortical dendritic morphology predict selective impairments in perceptual attentional set-shifting. *J Neurosci*. 2006;26(30):7870-4.
4. Radley JJ, Arias CM, Sawchenko PE. Regional differentiation of the medial prefrontal cortex in regulating adaptive responses to acute emotional stress. *J Neurosci*. 2006;26(50):12967-76.
5. Goldwater DS, Pavlides C, Hunter RG, Bloss EB, Hof PR, McEwen BS, Morrison JH. Structural and functional alterations to rat medial prefrontal cortex following chronic restraint stress and recovery. *Neuroscience*. 2009;164(2):798-808.
6. Hercher C, Canetti L, Turecki G, Mechawar N. Anterior cingulate pyramidal neurons display altered dendritic branching in depressed suicides. *J Psychiatr Res*. 2010;44(5):286-93.
7. Moda-Sava RN, Murdock MH, Parekh PK, Fetcho RN, Huang BS, Huynh TN, et al. Sustained rescue of prefrontal circuit dysfunction by antidepressant-induced spine formation. *Science*. 2019;364(6436).
8. Shansky RM, Hamo C, Hof PR, McEwen BS, Morrison JH. Stress-induced dendritic remodeling in the prefrontal cortex is circuit specific. *Cereb Cortex*. 2009;19(10):2479-84.
9. Fillinger C, Yalcin I, Barrot M, Veinante P. Afferents to anterior cingulate areas 24a and 24b and midcingulate areas 24a' and 24b' in the mouse. *Brain Struct Funct*. 2017;222(3):1509-32.
10. Anastasiades PG, Carter AG. Circuit organization of the rodent medial prefrontal cortex. *Trends Neurosci*. 2021;44(7):550-63.
11. Covington HE, 3rd, Lobo MK, Maze I, Vialou V, Hyman JM, Zaman S, et al. Antidepressant effect of optogenetic stimulation of the medial prefrontal cortex. *J Neurosci*. 2010;30(48):16082-90.
12. Chaudhury D, Walsh JJ, Friedman AK, Juarez B, Ku SM, Koo JW, et al. Rapid regulation of depression-related behaviours by control of midbrain dopamine neurons. *Nature*. 2013;493(7433):532-6.
13. Friedman AK, Walsh JJ, Juarez B, Ku SM, Chaudhury D, Wang J, et al. Enhancing depression mechanisms in midbrain dopamine neurons achieves homeostatic resilience. *Science*. 2014;344(6181):313-9.
14. Carreno FR, Donegan JJ, Boley AM, Shah A, DeGuzman M, Frazer A, Lodge DJ. Activation of a ventral hippocampus-medial prefrontal cortex pathway is both necessary and sufficient for an antidepressant response to ketamine. *Mol Psychiatry*. 2016;21(9):1298-308.
15. Felix-Ortiz AC, Burgos-Robles A, Bhagat ND, Leppla CA, Tye KM. Bidirectional modulation of anxiety-related and social behaviors by amygdala projections to the medial prefrontal cortex. *Neuroscience*. 2016;321:197-209.

16. Burgos-Robles A, Kimchi EY, Izadmehr EM, Porzenheim MJ, Ramos-Guasp WA, Nieh EH, et al. Amygdala inputs to prefrontal cortex guide behavior amid conflicting cues of reward and punishment. *Nat Neurosci.* 2017;20(6):824-35.
17. Klavir O, Prigge M, Sarel A, Paz R, Yizhar O. Manipulating fear associations via optogenetic modulation of amygdala inputs to prefrontal cortex. *Nat Neurosci.* 2017;20(6):836-44.
18. Sotres-Bayon F, Sierra-Mercado D, Pardilla-Delgado E, Quirk GJ. Gating of fear in prelimbic cortex by hippocampal and amygdala inputs. *Neuron.* 2012;76(4):804-12.
19. Adhikari A, Topiwala MA, Gordon JA. Synchronized activity between the ventral hippocampus and the medial prefrontal cortex during anxiety. *Neuron.* 2010;65(2):257-69.
20. Adhikari A, Topiwala MA, Gordon JA. Single units in the medial prefrontal cortex with anxiety-related firing patterns are preferentially influenced by ventral hippocampal activity. *Neuron.* 2011;71(5):898-910.
21. Padilla-Coreano N, Bolkan SS, Pierce GM, Blackman DR, Hardin WD, Garcia-Garcia AL, et al. Direct Ventral Hippocampal-Prefrontal Input Is Required for Anxiety-Related Neural Activity and Behavior. *Neuron.* 2016;89(4):857-66.
22. Padilla-Coreano N, Canetta S, Mikofsky RM, Alway E, Passecker J, Myroshnychenko MV, et al. Hippocampal-Prefrontal Theta Transmission Regulates Avoidance Behavior. *Neuron.* 2019;104(3):601-10 e4.
23. Woo E, Sansing LH, Arnsten AFT, Datta D. Chronic Stress Weakens Connectivity in the Prefrontal Cortex: Architectural and Molecular Changes. *Chronic Stress (Thousand Oaks).* 2021;5:24705470211029254.
24. Little JP, Carter AG. Subcellular synaptic connectivity of layer 2 pyramidal neurons in the medial prefrontal cortex. *J Neurosci.* 2012;32(37):12808-19.
25. Little JP, Carter AG. Synaptic mechanisms underlying strong reciprocal connectivity between the medial prefrontal cortex and basolateral amygdala. *J Neurosci.* 2013;33(39):15333-42.
26. Liu X, Carter AG. Ventral Hippocampal Inputs Preferentially Drive Corticocortical Neurons in the Infralimbic Prefrontal Cortex. *J Neurosci.* 2018;38(33):7351-63.
27. Herry C, Ciocchi S, Senn V, Demmou L, Muller C, Luthi A. Switching on and off fear by distinct neuronal circuits. *Nature.* 2008;454(7204):600-6.
28. Sanchez-Bellot C, AlSubaie R, Mishchanchuk K, Wee RWS, MacAskill AF. Two opposing hippocampus to prefrontal cortex pathways for the control of approach and avoidance behaviour. *Nat Commun.* 2022;13(1):339.
29. Choi JH, Sim SE, Kim JI, Choi DI, Oh J, Ye S, et al. Interregional synaptic maps among engram cells underlie memory formation. *Science.* 2018;360(6387):430-5.
30. Fuster J. *The Prefrontal Cortex.* 5th ed. Burlington: Elsevier Science; 2015. 1 online resource (461 p.) p.
31. Le Merre P, Ahrlund-Richter S, Carlen M. The mouse prefrontal cortex: Unity in diversity. *Neuron.* 2021;109(12):1925-44.
32. Schuman B, Machold RP, Hashikawa Y, Fuzik J, Fishell GJ, Rudy B. Four Unique Interneuron Populations Reside in Neocortical Layer 1. *J Neurosci.* 2019;39(1):125-39.

33. Han Y, Kebschull JM, Campbell RAA, Cowan D, Imhof F, Zador AM, Mrsic-Flogel TD. The logic of single-cell projections from visual cortex. *Nature*. 2018;556(7699):51-6.
34. Feldmeyer D. Excitatory neuronal connectivity in the barrel cortex. *Front Neuroanat*. 2012;6:24.
35. Sun W, Tan Z, Mensh BD, Ji N. Thalamus provides layer 4 of primary visual cortex with orientation- and direction-tuned inputs. *Nat Neurosci*. 2016;19(2):308-15.
36. Baker A, Kalmbach B, Morishima M, Kim J, Juavinett A, Li N, Dembrow N. Specialized Subpopulations of Deep-Layer Pyramidal Neurons in the Neocortex: Bridging Cellular Properties to Functional Consequences. *J Neurosci*. 2018;38(24):5441-55.
37. Constantinople CM, Bruno RM. Deep cortical layers are activated directly by thalamus. *Science*. 2013;340(6140):1591-4.
38. Lein ES, Hawrylycz MJ, Ao N, Ayres M, Bensinger A, Bernard A, et al. Genome-wide atlas of gene expression in the adult mouse brain. *Nature*. 2007;445(7124):168-76.
39. Yamawaki N, Borges K, Suter BA, Harris KD, Shepherd GM. A genuine layer 4 in motor cortex with prototypical synaptic circuit connectivity. *Elife*. 2014;3:e05422.
40. Wang Q, Ding SL, Li Y, Royall J, Feng D, Lesnar P, et al. The Allen Mouse Brain Common Coordinate Framework: A 3D Reference Atlas. *Cell*. 2020;181(4):936-53 e20.
41. Tasic B, Yao Z, Graybiuck LT, Smith KA, Nguyen TN, Bertagnolli D, et al. Shared and distinct transcriptomic cell types across neocortical areas. *Nature*. 2018;563(7729):72-8.
42. Cheriyan J, Kaushik MK, Ferreira AN, Sheets PL. Specific Targeting of the Basolateral Amygdala to Projectionally Defined Pyramidal Neurons in Prelimbic and Infralimbic Cortex. *eNeuro*. 2016;3(2).
43. Petreanu L, Huber D, Sobczyk A, Svoboda K. Channelrhodopsin-2-assisted circuit mapping of long-range callosal projections. *Nat Neurosci*. 2007;10(5):663-8.
44. Petreanu L, Mao T, Sternson SM, Svoboda K. The subcellular organization of neocortical excitatory connections. *Nature*. 2009;457(7233):1142-5.
45. Gokce O, Bonhoeffer T, Scheuss V. Clusters of synaptic inputs on dendrites of layer 5 pyramidal cells in mouse visual cortex. *Elife*. 2016;5.
46. Lafourcade M, van der Goes MH, Vardalaki D, Brown NJ, Voigts J, Yun DH, et al. Differential dendritic integration of long-range inputs in association cortex via subcellular changes in synaptic AMPA-to-NMDA receptor ratio. *Neuron*. 2022;110(9):1532-46 e4.
47. Arnsten AF. Stress signalling pathways that impair prefrontal cortex structure and function. *Nat Rev Neurosci*. 2009;10(6):410-22.
48. Sheline YI, Wang PW, Gado MH, Csernansky JG, Vannier MW. Hippocampal atrophy in recurrent major depression. *Proc Natl Acad Sci U S A*. 1996;93(9):3908-13.
49. Drevets WC, Price JL, Simpson JR, Jr., Todd RD, Reich T, Vannier M, Raichle ME. Subgenual prefrontal cortex abnormalities in mood disorders. *Nature*. 1997;386(6627):824-7.
50. Hamilton JP, Siemer M, Gotlib IH. Amygdala volume in major depressive disorder: a meta-analysis of magnetic resonance imaging studies. *Mol Psychiatry*. 2008;13(11):993-1000.



51. Siegle GJ, Thompson W, Carter CS, Steinhauer SR, Thase ME. Increased amygdala and decreased dorsolateral prefrontal BOLD responses in unipolar depression: related and independent features. *Biol Psychiatry*. 2007;61(2):198-209.
52. Davey CG, Whittle S, Harrison BJ, Simmons JG, Byrne ML, Schwartz OS, Allen NB. Functional brain-imaging correlates of negative affectivity and the onset of first-episode depression. *Psychol Med*. 2015;45(5):1001-9.
53. Geng H, Wu F, Kong L, Tang Y, Zhou Q, Chang M, et al. Disrupted Structural and Functional Connectivity in Prefrontal-Hippocampus Circuitry in First-Episode Medication-Naïve Adolescent Depression. *PLoS One*. 2016;11(2):e0148345.
54. Magalhaes R, Barriere DA, Novais A, Marques F, Marques P, Cerqueira J, et al. The dynamics of stress: a longitudinal MRI study of rat brain structure and connectome. *Mol Psychiatry*. 2018;23(10):1998-2006.
55. Gong Q, He Y. Depression, neuroimaging and connectomics: a selective overview. *Biol Psychiatry*. 2015;77(3):223-35.
56. Zhuo C, Li G, Lin X, Jiang D, Xu Y, Tian H, et al. The rise and fall of MRI studies in major depressive disorder. *Transl Psychiatry*. 2019;9(1):335.
57. Oh SW, Harris JA, Ng L, Winslow B, Cain N, Mihalas S, et al. A mesoscale connectome of the mouse brain. *Nature*. 2014;508(7495):207-14.
58. Muir J, Lopez J, Bagot RC. Wiring the depressed brain: optogenetic and chemogenetic circuit interrogation in animal models of depression. *Neuropsychopharmacology*. 2019;44(6):1013-26.
59. Einarsson EÖ, Flores A, Jercog D, Herry C. Basal amygdala inputs to the medial prefrontal cortex mediate fear memory strengthening. *bioRxiv*. 2022:2022.01.20.477064.
60. Becker LJ, Fillinger C, Waegaert R, Journee SH, Hener P, Ayazgok B, et al. The basolateral amygdala-anterior cingulate pathway contributes to depression-like behaviors and comorbidity with chronic pain behaviors in male mice. *Nat Commun*. 2023;14(1):2198.
61. Silverstein SE, O'Sullivan R, Bukalo O, Pati D, Schaffer JA, Limoges A, et al. A distinct cortical code for socially learned threat. *Nature*. 2024;626(8001):1066-72.
62. Parfitt GM, Nguyen R, Bang JY, Aqrabawi AJ, Tran MM, Seo DK, et al. Bidirectional Control of Anxiety-Related Behaviors in Mice: Role of Inputs Arising from the Ventral Hippocampus to the Lateral Septum and Medial Prefrontal Cortex. *Neuropsychopharmacology*. 2017;42(8):1715-28.
63. Wilke SA, Lavi K, Byeon S, Donohue KC, Sohal VS. Convergence of Clinically Relevant Manipulations on Dopamine-Regulated Prefrontal Activity Underlying Stress Coping Responses. *Biol Psychiatry*. 2022;91(9):810-20.
64. Song Q, Wei A, Xu H, Gu Y, Jiang Y, Dong N, et al. An ACC-VTA-ACC positive-feedback loop mediates the persistence of neuropathic pain and emotional consequences. *Nat Neurosci*. 2024;27(2):272-85.
65. Zhong P, Qin L, Yan Z. Dopamine Differentially Regulates Response Dynamics of Prefrontal Cortical Principal Neurons and Interneurons to Optogenetic Stimulation of Inputs from Ventral Tegmental Area. *Cereb Cortex*. 2020;30(8):4402-9.
66. Ellwood IT, Patel T, Wadia V, Lee AT, Liptak AT, Bender KJ, Sohal VS. Tonic or Phasic Stimulation of Dopaminergic Projections to Prefrontal Cortex Causes Mice to Maintain or Deviate from Previously Learned Behavioral Strategies. *J Neurosci*. 2017;37(35):8315-29.

67. Perez-Lopez JL, Contreras-Lopez R, Ramirez-Jarquin JO, Tecuapetla F. Direct Glutamatergic Signaling From Midbrain Dopaminergic Neurons Onto Pyramidal Prefrontal Cortex Neurons. *Front Neural Circuits*. 2018;12:70.
68. Post RJ, Warden MR. Depression: the search for separable behaviors and circuits. *Curr Opin Neurobiol*. 2018;49:192-200.
69. Phillips ML, Robinson HA, Pozzo-Miller L. Ventral hippocampal projections to the medial prefrontal cortex regulate social memory. *Elife*. 2019;8.
70. Lefebvre JL, Sanes JR, Kay JN. Development of dendritic form and function. *Annu Rev Cell Dev Biol*. 2015;31:741-77.
71. Holtmaat A, Svoboda K. Experience-dependent structural synaptic plasticity in the mammalian brain. *Nat Rev Neurosci*. 2009;10(9):647-58.
72. Branco T, Hausser M. The single dendritic branch as a fundamental functional unit in the nervous system. *Curr Opin Neurobiol*. 2010;20(4):494-502.
73. Berry KP, Nedivi E. Spine Dynamics: Are They All the Same? *Neuron*. 2017;96(1):43-55.
74. Nagerl UV, Kostinger G, Anderson JC, Martin KA, Bonhoeffer T. Protracted synaptogenesis after activity-dependent spinogenesis in hippocampal neurons. *J Neurosci*. 2007;27(30):8149-56.
75. Zito K, Scheuss V, Knott G, Hill T, Svoboda K. Rapid functional maturation of nascent dendritic spines. *Neuron*. 2009;61(2):247-58.
76. Citri A, Malenka RC. Synaptic plasticity: multiple forms, functions, and mechanisms. *Neuropsychopharmacology*. 2008;33(1):18-41.
77. Malenka RC, Bear MF. LTP and LTD: an embarrassment of riches. *Neuron*. 2004;44(1):5-21.
78. Nakahata Y, Yasuda R. Plasticity of Spine Structure: Local Signaling, Translation and Cytoskeletal Reorganization. *Front Synaptic Neurosci*. 2018;10:29.
79. Stein IS, Zito K. Dendritic Spine Elimination: Molecular Mechanisms and Implications. *Neuroscientist*. 2019;25(1):27-47.
80. Poirazi P, Brannon T, Mel BW. Pyramidal neuron as two-layer neural network. *Neuron*. 2003;37(6):989-99.
81. Polsky A, Mel BW, Schiller J. Computational subunits in thin dendrites of pyramidal cells. *Nat Neurosci*. 2004;7(6):621-7.
82. London M, Hausser M. Dendritic computation. *Annu Rev Neurosci*. 2005;28:503-32.
83. Larkum ME, Nevian T, Sandler M, Polsky A, Schiller J. Synaptic integration in tuft dendrites of layer 5 pyramidal neurons: a new unifying principle. *Science*. 2009;325(5941):756-60.
84. Gidon A, Zolnik TA, Fidzinski P, Bolduan F, Papoutsi A, Poirazi P, et al. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*. 2020;367(6473):83-7.
85. Hedrick NG, Harward SC, Hall CE, Murakoshi H, McNamara JO, Yasuda R. Rho GTPase complementation underlies BDNF-dependent homo- and heterosynaptic plasticity. *Nature*. 2016;538(7623):104-8.
86. Francioni V, Harnett MT. Rethinking Single Neuron Electrical Compartmentalization: Dendritic Contributions to Network Computation In Vivo. *Neuroscience*. 2022;489:185-99.

87. Otor Y, Achvat S, Cermak N, Benisty H, Abboud M, Barak O, et al. Dynamic compartmental computations in tuft dendrites of layer 5 neurons during motor behavior. *Science*. 2022;376(6590):267-75.
88. Fisek M, Herrmann D, Egea-Weiss A, Cloves M, Bauer L, Lee TY, et al. Cortico-cortical feedback engages active dendrites in visual cortex. *Nature*. 2023;617(7962):769-76.
89. Xu Z, Geron E, Perez-Cuesta LM, Bai Y, Gan WB. Generalized extinction of fear memory depends on co-allocation of synaptic plasticity in dendrites. *Nat Commun*. 2023;14(1):503.
90. Manita S, Ross WN. Synaptic activation and membrane potential changes modulate the frequency of spontaneous elementary Ca<sup>2+</sup> release events in the dendrites of pyramidal neurons. *J Neurosci*. 2009;29(24):7833-45.
91. Harvey CD, Yasuda R, Zhong H, Svoboda K. The spread of Ras activity triggered by activation of a single dendritic spine. *Science*. 2008;321(5885):136-40.
92. Losonczy A, Makara JK, Magee JC. Compartmentalized dendritic plasticity and input feature storage in neurons. *Nature*. 2008;452(7186):436-41.
93. Branco T, Clark BA, Hausser M. Dendritic discrimination of temporal input sequences in cortical neurons. *Science*. 2010;329(5999):1671-5.
94. Tran-Van-Minh A, Caze RD, Abrahamsson T, Cathala L, Gutkin BS, DiGregorio DA. Contribution of sublinear and supralinear dendritic integration to neuronal computations. *Front Cell Neurosci*. 2015;9:67.
95. Kastellakis G, Poirazi P. Synaptic Clustering and Memory Formation. *Front Mol Neurosci*. 2019;12:300.
96. Wilson DE, Whitney DE, Scholl B, Fitzpatrick D. Orientation selectivity and the functional clustering of synaptic inputs in primary visual cortex. *Nat Neurosci*. 2016;19(8):1003-9.
97. Xu NL, Harnett MT, Williams SR, Huber D, O'Connor DH, Svoboda K, Magee JC. Nonlinear dendritic integration of sensory and motor input during an active sensing task. *Nature*. 2012;492(7428):247-51.
98. Marti Mengual U, Wybo WAM, Spierenburg LJE, Santello M, Senn W, Nevian T. Efficient Low-Pass Dendro-Somatic Coupling in the Apical Dendrite of Layer 5 Pyramidal Neurons in the Anterior Cingulate Cortex. *J Neurosci*. 2020;40(46):8799-815.
99. Brandalise F, Kalmbach BE, Cook EP, Brager DH. Impaired dendritic spike generation in the Fragile X prefrontal cortex is due to loss of dendritic sodium channels. *J Physiol*. 2023;601(4):831-45.
100. Song C, Moyer JR, Jr. Layer- and subregion-specific differences in the neurophysiological properties of rat medial prefrontal cortex pyramidal neurons. *J Neurophysiol*. 2018;119(1):177-91.
101. Dembrow NC, Zemelman BV, Johnston D. Temporal dynamics of L5 dendrites in medial prefrontal cortex regulate integration versus coincidence detection of afferent inputs. *J Neurosci*. 2015;35(11):4501-14.
102. Kalmbach BE, Chitwood RA, Dembrow NC, Johnston D. Dendritic generation of mGluR-mediated slow afterdepolarization in layer 5 neurons of prefrontal cortex. *J Neurosci*. 2013;33(33):13518-32.
103. Seong HJ, Carter AG. D1 receptor modulation of action potential firing in a subpopulation of layer 5 pyramidal neurons in the prefrontal cortex. *J Neurosci*. 2012;32(31):10516-21.

104. Gee S, Ellwood I, Patel T, Luongo F, Deisseroth K, Sohal VS. Synaptic activity unmasks dopamine D2 receptor modulation of a specific class of layer V pyramidal neurons in prefrontal cortex. *J Neurosci*. 2012;32(14):4959-71.
105. Moberg S, Takahashi N. Neocortical layer 5 subclasses: From cellular properties to roles in behavior. *Front Synaptic Neurosci*. 2022;14:1006773.
106. Wang M, Ramos BP, Paspalas CD, Shu Y, Simen A, Duque A, et al. Alpha2A-adrenoceptors strengthen working memory networks by inhibiting cAMP-HCN channel signaling in prefrontal cortex. *Cell*. 2007;129(2):397-410.
107. Radley JJ, Sisti HM, Hao J, Rocher AB, McCall T, Hof PR, et al. Chronic behavioral stress induces apical dendritic reorganization in pyramidal neurons of the medial prefrontal cortex. *Neuroscience*. 2004;125(1):1-6.
108. Cerqueira JJ, Mailliet F, Almeida OF, Jay TM, Sousa N. The prefrontal cortex as a key target of the maladaptive response to stress. *J Neurosci*. 2007;27(11):2781-7.
109. Hughes BW, Siemsen BM, Tsvetkov E, Berto S, Kumar J, Cornbrooks RG, et al. NPAS4 in the medial prefrontal cortex mediates chronic social defeat stress-induced anhedonia-like behavior and reductions in excitatory synapses. *Elife*. 2023;12.
110. Li N, Lee B, Liu RJ, Banasr M, Dwyer JM, Iwata M, et al. mTOR-dependent synapse formation underlies the rapid antidepressant effects of NMDA antagonists. *Science*. 2010;329(5994):959-64.
111. Duman RS, Aghajanian GK. Synaptic dysfunction in depression: potential therapeutic targets. *Science*. 2012;338(6103):68-72.
112. Phoumthipphavong V, Barthas F, Hassett S, Kwan AC. Longitudinal Effects of Ketamine on Dendritic Architecture In Vivo in the Mouse Medial Frontal Cortex. *eNeuro*. 2016;3(2).
113. Ali F, Gerhard DM, Sweasy K, Pothula S, Pittenger C, Duman RS, Kwan AC. Ketamine disinhibits dendrites and enhances calcium signals in prefrontal dendritic spines. *Nat Commun*. 2020;11(1):72.
114. Shao LX, Liao C, Gregg I, Davoudian PA, Savalia NK, Delagarza K, Kwan AC. Psilocybin induces rapid and persistent growth of dendritic spines in frontal cortex in vivo. *Neuron*. 2021;109(16):2535-44 e4.
115. Savalia NK, Shao LX, Kwan AC. A Dendrite-Focused Framework for Understanding the Actions of Ketamine and Psychedelics. *Trends Neurosci*. 2021;44(4):260-75.
116. Liu WZ, Zhang WH, Zheng ZH, Zou JX, Liu XX, Huang SH, et al. Identification of a prefrontal cortex-to-amygdala pathway for chronic stress-induced anxiety. *Nat Commun*. 2020;11(1):2221.
117. Liu WZ, Wang CY, Wang Y, Cai MT, Zhong WX, Liu T, et al. Circuit- and laminar-specific regulation of medial prefrontal neurons by chronic stress. *Cell Biosci*. 2023;13(1):90.
118. Golden SA, Covington HE, 3rd, Berton O, Russo SJ. A standardized protocol for repeated social defeat stress in mice. *Nat Protoc*. 2011;6(8):1183-91.
119. Goodwin NL, Choong JJ, Hwang S, Pitts K, Bloom L, Islam A, et al. Simple Behavioral Analysis (SimBA) as a platform for explainable machine learning in behavioral neuroscience. *Nat Neurosci*. 2024;27(7):1411-24.
120. De Kloet ER, Vreugdenhil E, Oitzl MS, Joels M. Brain corticosteroid receptor balance in health and disease. *Endocr Rev*. 1998;19(3):269-301.
121. McEwen BS. Protective and damaging effects of stress mediators. *N Engl J Med*. 1998;338(3):171-9.

122. Liston C, Cichon JM, Jeanneteau F, Jia Z, Chao MV, Gan WB. Circadian glucocorticoid oscillations promote learning-dependent synapse formation and maintenance. *Nat Neurosci*. 2013;16(6):698-705.
123. Kim HD, Wei J, Call T, Quintus NT, Summers AJ, Carotenuto S, et al. Shisa6 mediates cell-type specific regulation of depression in the nucleus accumbens. *Mol Psychiatry*. 2021;26(12):7316-27.
124. Neher RA, Mitkovski M, Kirchhoff F, Neher E, Theis FJ, Zeug A. Blind source separation techniques for the decomposition of multiply labeled fluorescence images. *Biophys J*. 2009;96(9):3791-800.
125. Majumdar A, Cesario WC, White-Grindley E, Jiang H, Ren F, Khan MR, et al. Critical role of amyloid-like oligomers of *Drosophila* Orb2 in the persistence of memory. *Cell*. 2012;148(3):515-29.
126. Choi DI, Kaang BK. Interrogating structural plasticity among synaptic engrams. *Curr Opin Neurobiol*. 2022;75:102552.
127. McRae TD, Oleksyn D, Miller J, Gao YR. Robust blind spectral unmixing for fluorescence microscopy using unsupervised learning. *PLoS One*. 2019;14(12):e0225410.
128. Jimenez-Sanchez D, Ariz M, Morgado JM, Cortes-Dominguez I, Ortiz-de-Solorzano C. NMF-RI: blind spectral unmixing of highly mixed multispectral flow and image cytometry data. *Bioinformatics*. 2020;36(5):1590-8.
129. Maric D, Jahanipour J, Li XR, Singh A, Mobiny A, Van Nguyen H, et al. Whole-brain tissue mapping toolkit using large-scale highly multiplexed immunofluorescence imaging and deep neural networks. *Nat Commun*. 2021;12(1):1550.
130. Seo J, Sim Y, Kim J, Kim H, Cho I, Nam H, et al. PICASSO allows ultra-multiplexed fluorescence imaging of spatially overlapping proteins without reference spectra measurements. *Nat Commun*. 2022;13(1):2475.
131. Santuy A, Rodriguez JR, DeFelipe J, Merchan-Perez A. Study of the Size and Shape of Synapses in the Juvenile Rat Somatosensory Cortex with 3D Electron Microscopy. *eNeuro*. 2018;5(1).
132. Masters BR. Concepts and Criteria of Resolution. In: Masters BR, editor. *Superresolution Optical Microscopy: The Quest for Enhanced Resolution and Contrast*. Cham: Springer International Publishing; 2020. p. 13-32.
133. Horl D, Rojas Rusak F, Preusser F, Tillberg P, Randel N, Chhetri RK, et al. BigStitcher: reconstructing high-resolution image datasets of cleared and expanded samples. *Nat Methods*. 2019;16(9):870-4.
134. Perdigao LMA, Berger C, Yee NB, Darrow MC, Basham M. RedLionfish - fast Richardson-Lucy Deconvolution package for efficient point spread function suppression in volumetric data. *Wellcome Open Res*. 2024;9:296.
135. Dougherty B. Diffraction PSF 3D [Internet] 2005 [updated 18.10.2024]. Available from: <https://www.optinav.info/Diffraction-PSF-3D.htm>.
136. Gliko O, Mallory M, Dalley R, Gala R, Gornet J, Zeng H, et al. High-throughput analysis of dendrite and axonal arbors reveals transcriptomic correlates of neuroanatomy. *Nat Commun*. 2024;15(1):6337.
137. Motta A, Berning M, Boergens KM, Staffler B, Beining M, Loomba S, et al. Dense connectomic reconstruction in layer 4 of the somatosensory cortex. *Science*. 2019;366(6469).

138. Motta A, Schurr M, Staffler B, Helmstaedter M. Big data in nanoscale connectomics, and the greed for training labels. *Curr Opin Neurobiol.* 2019;55:180-7.
139. Danielson E, Lee SH. SynPAnal: software for rapid quantification of the density and intensity of protein puncta from fluorescence microscopy images of neurons. *PLoS One.* 2014;9(12):e115298.
140. Iascone DM, Li Y, Sumbul U, Doron M, Chen H, Andreu V, et al. Whole-Neuron Synaptic Mapping Reveals Spatially Precise Excitatory/Inhibitory Balance Limiting Dendritic and Somatic Spiking. *Neuron.* 2020;106(4):566-78 e8.
141. Wang Y, Wang C, Ranefall P, Broussard GJ, Wang Y, Shi G, et al. SynQuant: an automatic tool to quantify synapses from microscopy images. *Bioinformatics.* 2020;36(5):1599-606.
142. Rössler W, Spaethe J, Groh C. Pitfalls of using confocal-microscopy based automated quantification of synaptic complexes in honeybee mushroom bodies (response to Peng and Yang 2016). *Sci Rep.* 2017;7(1):9786.
143. Kuhn HW. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly.* 1955;2(1-2):83-97.
144. McInnes L, Healy J, Melville J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:180203426.* 2018.
145. Armitage P, Berry G. *Statistical methods in medical research.* 3rd ed. Oxford ; Boston: Blackwell Scientific Publications; 1994. xi, 620 pages p.
146. Sesack SR, Deutch AY, Roth RH, Bunney BS. Topographical organization of the efferent projections of the medial prefrontal cortex in the rat: an anterograde tract-tracing study with *Phaseolus vulgaris leucoagglutinin*. *J Comp Neurol.* 1989;290(2):213-42.
147. McGarry LM, Carter AG. Prefrontal Cortex Drives Distinct Projection Neurons in the Basolateral Amygdala. *Cell Rep.* 2017;21(6):1426-33.
148. Tanimizu T, Kenney JW, Okano E, Kadoma K, Frankland PW, Kida S. Functional Connectivity of Multiple Brain Regions Required for the Consolidation of Social Recognition Memory. *J Neurosci.* 2017;37(15):4103-16.
149. Hitti FL, Siegelbaum SA. The hippocampal CA2 region is essential for social memory. *Nature.* 2014;508(7494):88-92.
150. Okuyama T, Kitamura T, Roy DS, Itohara S, Tonegawa S. Ventral CA1 neurons store social memory. *Science.* 2016;353(6307):1536-41.
151. Lee E, Rhim I, Lee JW, Ghim JW, Lee S, Kim E, Jung MW. Enhanced Neuronal Activity in the Medial Prefrontal Cortex during Social Approach Behavior. *J Neurosci.* 2016;36(26):6926-36.
152. McGarry LM, Carter AG. Inhibitory Gating of Basolateral Amygdala Inputs to the Prefrontal Cortex. *J Neurosci.* 2016;36(36):9391-406.
153. Likhtik E, Stujenske JM, Topiwala MA, Harris AZ, Gordon JA. Prefrontal entrainment of amygdala activity signals safety in learned fear and innate anxiety. *Nat Neurosci.* 2014;17(1):106-13.
154. Kaidanovich-Beilin O, Lipina T, Vukobradovic I, Roder J, Woodgett JR. Assessment of social interaction behaviors. *J Vis Exp.* 2011(48).
155. Lissek S, Rabin S, Heller RE, Lukenbaugh D, Geraci M, Pine DS, Grillon C. Overgeneralization of conditioned fear as a pathogenic marker of panic disorder. *Am J Psychiatry.* 2010;167(1):47-55.
156. Jovanovic T, Kazama A, Bachevalier J, Davis M. Impaired safety signal learning may be a biomarker of PTSD. *Neuropharmacology.* 2012;62(2):695-704.

157. Lissek S, Kaczkurkin AN, Rabin S, Geraci M, Pine DS, Grillon C. Generalized anxiety disorder is associated with overgeneralization of classically conditioned fear. *Biol Psychiatry*. 2014;75(11):909-15.
158. Kaczkurkin AN, Burton PC, Chazin SM, Manbeck AB, Espensen-Sturges T, Cooper SE, et al. Neural Substrates of Overgeneralized Conditioned Fear in PTSD. *Am J Psychiatry*. 2017;174(2):125-34.
159. Fetcho RN, Hall BS, Estrin DJ, Walsh AP, Schuette PJ, Kaminsky J, et al. Regulation of social interaction in mice by a frontostriatal circuit modulated by established hierarchical relationships. *Nat Commun*. 2023;14(1):2487.
160. Grundemann J, Bitterman Y, Lu T, Krabbe S, Grewe BF, Schnitzer MJ, Luthi A. Amygdala ensembles encode behavioral states. *Science*. 2019;364(6437).
161. Martin-Fernandez M, Menegolla AP, Lopez-Fernandez G, Winke N, Jercog D, Kim HR, et al. Prefrontal circuits encode both general danger and specific threat representations. *Nat Neurosci*. 2023;26(12):2147-57.
162. Zelikowsky M, Hersman S, Chawla MK, Barnes CA, Fanselow MS. Neuronal ensembles in amygdala, hippocampus, and prefrontal cortex track differential components of contextual fear. *J Neurosci*. 2014;34(25):8462-6.
163. Hare BD, Shinohara R, Liu RJ, Pothula S, DiLeone RJ, Duman RS. Optogenetic stimulation of medial prefrontal cortex *Drd1* neurons produces rapid and long-lasting antidepressant effects. *Nat Commun*. 2019;10(1):223.
164. Tritsch NX, Sabatini BL. Dopaminergic modulation of synaptic transmission in cortex and striatum. *Neuron*. 2012;76(1):33-50.
165. Yizhar O, Fenno LE, Prigge M, Schneider F, Davidson TJ, O'Shea DJ, et al. Neocortical excitation/inhibition balance in information processing and social dysfunction. *Nature*. 2011;477(7363):171-8.
166. Sun Q, Li X, Ren M, Zhao M, Zhong Q, Ren Y, et al. A whole-brain map of long-range inputs to GABAergic interneurons in the mouse medial prefrontal cortex. *Nat Neurosci*. 2019;22(8):1357-70.
167. Lee AT, Cunniff MM, See JZ, Wilke SA, Luongo FJ, Ellwood IT, et al. VIP Interneurons Contribute to Avoidance Behavior by Regulating Information Flow across Hippocampal-Prefrontal Networks. *Neuron*. 2019;102(6):1223-34 e4.
168. Soumier A, Sibille E. Opposing effects of acute versus chronic blockade of frontal cortex somatostatin-positive inhibitory neurons on behavioral emotionality in mice. *Neuropsychopharmacology*. 2014;39(9):2252-62.
169. Perova Z, Delevich K, Li B. Depression of excitatory synapses onto parvalbumin interneurons in the medial prefrontal cortex in susceptibility to stress. *J Neurosci*. 2015;35(7):3201-6.
170. Harris AZ, Atsak P, Bretton ZH, Holt ES, Alam R, Morton MP, et al. A Novel Method for Chronic Social Defeat Stress in Female Mice. *Neuropsychopharmacology*. 2018;43(6):1276-83.

## 8. EKLER

### EK-1: Görüntü İşleme Sırasında Kullanılan Algoritmalar

```
def tiffmxnslicer(file, M, N):
    '''
    Elimizdeki bilgisayar büyük görüntülerin ayrıştırmasını bir
    seferde yapamayacağı için,
    görüntüler birleştirildikten sonra istenen sayıda veya
    boyutta görüntü grubuna kesilmesi hedeflenmiştir.

    Girdiler: file: .tif uzantılı görüntü ismi/ çalışma dosyası
    içinde bulunmalıdır. Veri tipi: string
               M: elde edilecek görüntü sayısının bir çarpanı.
Veri tipi: int
               N: elde edilecek görüntü sayısının bir çarpanı.
Veri tipi: int

    Çıktı: M x N sayıda çok spektrumlu görüntü. .tif uzantılı
    kaydedilir.
    '''

    import os # Kütüphanelerin yüklenmesi
    import numpy as np
    from skimage import io
    import sys

    bigtiff = io.imread(file) # Büyük .tif uzantılı görüntünün
    okunması

    if len(bigtiff.shape) == 4: # Multispektral görüntü spektral
    özelliklerine göre M x N parçaya ayrılır.
        Z, C, H, W = bigtiff.shape
        for m in np.arange(M, dtype=np.int8):
            for n in np.arange(N, dtype=np.int8):
                tiff_img =
bigtiff[:, :, round((H*m)/M):round((H*(m+1)/M)),
round((W*n)/N):round((W*(n+1)/N))]
                io.imsave('%s_%d_%d_of_%dx%d.tif' %(file, m, n,
M, N), tiff_img)
                print('Row: ', m, 'Column: ', n, ' with size of',
tiff_img.shape, 'saved as ', '%s_%d_%d_of_%dx%d.tif' %(file, m,
n, M, N))

    elif len(bigtiff.shape) == 3:
        Z, H, W = bigtiff.shape
```



```

        for m in np.arange(M, dtype=np.int8):
            for n in np.arange(N, dtype=np.int8):
                tiff_img =
bigtiff[:,round((H*m)/M):round((H*(m+1)/M)),
round((W*n)/N):round((W*(n+1)/N))]
                io.imsave('%s_%d_%d_of_%dx%d.tif' %(file, m, n,
M, N), tiff_img)
                print('Row: ', m, 'Column: ', n, ' with size of',
tiff_img.shape, 'saved as ', '%s_%d_%d_of_%dx%d.tif' %(file, m,
n, M, N))

```

```

def GPUdeconvolution(path_from_curdir, num_iter, sigma):

    '''

    Spektral ayrıştırılan sinyallerin pikselleri izometrik
    yapıldıktan sonra oluşturulan PSF (nokta dağılım fonksiyonu)
    kullanılarak Richardson-Lucy dekonvolüsyonundan geçirildi.
    Sinaptik sinyaller dekonvolüsyon sonrasında ayrıca Gauss fil-
    trelemeden geçirildi.

    Bu algoritma bir dosya içerisindeki tüm görüntüleri GPU
    yardımıyla ayrı ayrı dekonvolve eder ve işlem sonucunda oluşan
    dosyayı kaydeder.

    Girdiler: path_from_curdir: Ana dosyadan itibaren hedeflenen
    dosyaya uzanan hefef yolu
                num_iter: Dekonvolüsyon iterasyon sayısı
                sigma: Gauss operatörün standart sap-
    ması, Fijideki operasyonda çap terimine karşılık gelmektedir.

    Çıktı: Dekonvolve görüntü

    Richardson-Lucy dekonvolüsyon işlemi için açık erişimli
    'RedLionfish deconvolution' kütüphanesi kullanılmaktadır.
    Kütüphaneye ulaşmak için: https://github.com/rosalindfranklininstitute/RedLionfish

    '''

import os # Kütüphanelerin yüklenmesi
from skimage import io
import numpy as np
import RedLionfishDeconv as rl
from scipy.ndimage import gaussian_filter

```

```

path = os.getcwd()
main_path = os.path.join(path, path_from_curdir)
os.chdir(main_path) # Dosya işlem merkezinin .tif uzantılı
dosyaların bulunduğu bölgeye çekilmesi

file_list = os.listdir()
num_file = len(file_list)
start = 0
uint8_output = []

#print('Here1')

for file in file_list:

    if 'PSF' in file: #PSF görüntünün okunması; her renk flo-
resans için farklı sinyaller kullanıldı
        psf = io.imread(file)
        psf_file = file

#print(psf)

for file in file_list:

    if 'PSF' in file:
        continue

    if '.txt' in file:
        continue

    if '_dend_' in file:
        print(file)

        img = io.imread(file)

        uint8_output = True # Dendritik sinyallere ek gauss
filtreleme uygulanmadı. Oluşan görüntü doğrudan 8 bit kaydedildi
        data_type = np.uint8
        exc = 585

    elif '_cyan_' in file:# Sinaptik sinyallere ek gauss fil-
treleme uygulanacağı için görüntüler önce 32bit elde edildi.
        # Gauss filtreleme sonrasında
görüntüler 8 bit görüntüye çevrilerek kaydedildi.

        img = io.imread(file)

```

```

        uint8_output = False
        data_type = np.float32
        exc = 480

    elif '_green_' in file:

        img = io.imread(file)

        uint8_output = False
        data_type = np.float32
        exc = 509

    elif '_yellow_' in file:

        img = io.imread(file)

        uint8_output = False
        data_type = np.float32
        exc = 527

    else:
        continue

    print('Deconvolving: ', file)
    print('with PSF: ', psf_file)
    print('will NOT do Gaussian', uint8_output)

    img_decv = rl.doRLDeconvolutionFromNpArrays(img, # Dekon-
volüsyon
                                                psf,
                                                niter =
num_iter,
                                                method =
'gpu',
                                                resAsUInt8 =
uint8_output)

    if uint8_output == False:
        print('Gaussian filtering...') # Gauss filtreleme
        img_decv_gauss = gaussian_filter(img_decv, sigma =
sigma)
        img_decv_gauss_scale =
np.round((img_decv_gauss/np.max(img_decv_gauss)) * 255)
        img_decv_gauss_scale_8bit =
img_decv_gauss_scale.astype(np.uint8)

```

```

        print('Saving...')
        io.imsave('%s_deconv_%d_its_gauss%d_8bit.tif' %(file,
num_iter, sigma), img_decv_gauss_scale_8bit)

    else:

        print('Saving...') # Dosyaların kaydedilmesi
        io.imsave('%s_RL_%dits_8bit.tif' %(file, num_iter),
img_decv)

        start = start + 1
        print('Deconvolution: ', start, '/', num_file-1, ' is
done...')

    os.chdir(path)

```

```

def swcslicer(filepath_from_main, syn_thr, dend_thr, num_of_sub-
slice):

    '''
        Gliko ve arkadaşlarının segmentasyon ve rekonstrüksiyon yönt-
emiyle elde segmente dendritik görüntü ve rekonstrüksiyon ağacı,
        Vaa3D yazılımı içindeki Synapse Detector algoritmasının
        verimli çalışması için uygun boyuttaki görüntü ve rekonstrüksiyon
        parçalarına bölünür.
        Burada önemli olan rekonstrüksiyon ağacı ile segmente dendrit
        görüntülerinin çakışmasının korunumunun sağlanmasıdır.

        Kolaylık olması açısından dekonvolve edilen görüntü parçaları
        kılavuz olması için kullanılmaktadır.
        Eğer bir kortikal kolon M parçaya ayrılmışsa ve her parça da
        N bölgeye ayrılacaksa elimizdeki dendritik rekonstrüksiyon ağacı
        M x N bölgeye ayrılacaktır
        Sonrasında Synapse Detector algoritması her bir renk için M x
        N kez çalıştırılacaktır.

        Girdiler: filepath_from_main: Ana dosyadan itibaren
        hedeflenen dosyaya uzanan hedef yolu
                    syn_thr: Sinaps görüntü parçalarına uy-
gulanacak eşik değerleri içeren vektör, veri tipi: uint8;
                    dend_thr: Dendritik segmentasyon
görüntüsüne uygulanacak eşik değer, veri tipi: uint8
                    num_of_subslice: Bir kortikal kolon görüntü par-
çasının ayrılacağı bölge sayısı
    '''

```

Çıktılar: .tif uzantılı M x N tane dendritik segmentasyon görüntüsü, veri tipi uint8 RGB görüntü  
 .swc uzantılı M x N tane dendritik rekonstrüksiyon ağacı

```
'''

import os # Kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd
from skimage import io

curr_dir = os.getcwd()# Dosya işlem merkezinin .tif ve .swc
uzantılı dosyaların bulunduğu bölgeye çekilmesi
file_path = os.path.join(curr_dir, filepath_from_main)
os.chdir(file_path)

file_list = os.listdir()
syn_thr = np.repeat(syn_thr, num_of_subslice)

img_file_names = []
z_list = []
y_list = 0
x_list = 0
x_coor = 0
y_coor = 0

for file in file_list:
    if '.swc' in file:# Rekonstrüksiyon ağacının okunması
        print('reconst_file', file)
        reconst_filename = file
        reconst = pd.read_csv(reconst_filename, sep = ' ',
header = 0)

        elif '_segmentation.tif' in file: # Dendritik segmenta-
syon görüntüsünün okunması
            print('dendrite_file', file)
            tileimg_filename = file
            tile_img = io.imread(tileimg_filename)
            Z_merged, Y_merged, X_merged = tile_img.shape

            elif '_merged.tif' in file: # Birleştirilmiş sinaps
görüntüsünün okunması
                print('synapse file', file)
                color_tileimg_filename = file
                color_tile_img = io.imread(color_tileimg_filename)
```

```

        Zc_merged, Yc_merged, Xc_merged =
color_tile_img.shape

        else: # Dendritik rekonstrüksiyon ağacının kesilmesinde
kılavuz olacak dendritik görüntü parçalarının okunması
            print('part file', file)
            part_filename = file
            part_img = io.imread(part_filename)
            Z_part, Y_part, X_part = part_img.shape
            for i in np.arange(num_of_subslice):
                x_coor += np.round(X_part/num_of_subslice)
                y_coor += np.round(Y_part/num_of_subslice)
                x_list = np.append(x_list, x_coor)
                y_list = np.append(y_list, y_coor)

del part_img
x_list = x_list.astype(np.uint)
y_list = y_list.astype(np.uint)

        if X_merged > Y_merged:# Kiremit tarama görüntülemesinin
yönüne (yatay veya dikey) göre koordinatların düzenlenmesi

            print('x points using', x_list)

            for i in np.arange(len(x_list) - 1):
                img_reconst_place = np.where((reconst['x'] >
x_list[i]) & # Dendritik rekonstrüksiyonun kesilmesi
                                                (reconst['x'] <=
x_list[i+1]))

                img_reconst = reconst.iloc[img_reconst_place]
                img_reconst['x'] = img_reconst['x'] - x_list[i]

                new_img = tile_img[:, :, x_list[i]: x_list[i+1]]
                new_img = np.flip(new_img, axis = 1)
                new_img[np.where(new_img >= dend_thr)] = 255 # Den-
dritik görüntülere eşikleme uygulanması
                new_img[np.where(new_img < dend_thr)] = 0

                new_color_img = color_tile_img[:, :, x_list[i]:
x_list[i+1]]
                new_color_img = np.flip(new_color_img, axis = 1) #
Sinaptik görüntülere eşikleme uygulanması
                new_color_img[np.where(new_color_img >= syn_thr[i])]
= 255

```

```

        new_color_img[np.where(new_color_img < syn_thr[i])] =
0
    new_img[np.where(new_color_img == 255)] = 255

    img_blue = np.zeros(new_img.shape, dtype = np.uint8)
# Oluşturulacak RGB görüntüdeki mavi katman tespit edilen sin-
# apsların

# kaydedilmesi amacıyla 0 matrisi olarak düzenlendi

    img_merged = np.concatenate((new_img[...],
np.newaxis],
                                new_color_img[...],
np.newaxis],
                                img_blue[...],
np.newaxis]),
                                axis = 3)

    img_merged = np.transpose(img_merged, (0, 1, 2, 3)) #
RGB görüntünün oluşturulması

    print('Saving file #', i) # Görüntülerin ve
rekonstrüksiyonların kaydedilmesi
    img_reconst.to_csv('%s_reconst_%d.swc' %(re-
const_filename[:-4], i), sep = ' ', index = False)
    tiff.imsave('%s_%d.tif' %(tileimg_filename[:-4], i),
img_merged)

    if Y_merged > X_merged:

        print('y points using', y_list)

        for i in np.arange(len(x_list) - 1):
            img_reconst_place = np.where((reconst['y'] >
y_list[i]) & # Dendiritk rekonstrüksiyonun kesilmesi
                                           (reconst['y'] <=
y_list[i+1]))

            img_reconst = reconst.iloc[img_reconst_place]
            img_reconst['y'] = img_reconst['y'] - y_list[i]

            new_img = tile_img[:, y_list[i]: y_list[i+1], :]
            new_img = np.flip(new_img, axis = 1)

```

```

        new_img[np.where(new_img >= dend_thr)] = 255 # Den-
dritik görüntülere eşikleme uygulanması
        new_img[np.where(new_img < dend_thr)] = 0

        new_color_img = color_tile_img[:, y_list[i]:
y_list[i+1], :]
        new_color_img = np.flip(new_color_img, axis = 1)
        new_color_img[np.where(new_color_img >= syn_thr[i])]
= 255 # Sinaptik görüntülere eşikleme uygulanması
        new_color_img[np.where(new_color_img < syn_thr[i])] =
0

        new_img[np.where(new_color_img == 255)] = 255
        img_blue = np.zeros(new_img.shape, dtype = np.uint8) #
Oluşturulacak RGB görüntüdeki mavi katman tespit edilen sin-
apsların

# kaydedilmesi amacıyla 0 matrisi olarak düzenlendi

        img_merged = np.concatenate((new_img[...],
np.newaxis],
                                     new_color_img[...],
np.newaxis],
                                     img_blue[...],
np.newaxis]),
                                     axis = 3)

        img_merged = np.transpose(img_merged, (0, 1, 2, 3)) #
RGB görüntünün oluşturulması

        print('Saving file #', i) # Görüntülerin ve
rekonstrüksiyonların kaydedilmesi
        img_reconst.to_csv('%s_reconst_%d.swc' %(re-
const_filename[:-4], i), sep = ' ', index = False)
        tiff.imwrite('%s_%d.tif' %(tileimg_filename[:-4], i),
img_merged)

os.chdir(curr_dir)

```



```

def get_real_nodes_to_isq(reconst_file_name, synapse_file_name):
    '''
        Synapse Detector algoritması tespit edilen sinapsa en yakın
        .swc nodu doğrudan doğruya numarasını vermek yerine listedeki
        sırasını vermektedir.
        Bu algoritma tespit edilen sinapsa en yakın nodun doğru
        numarasını bulmak için hazırlandı.

        Girdiler: reconst_file_name: Rekonstrüksiyon dosya ismi
                 synapse_file_name: Tespit edilen sinapsların dosya
        ismi

        Çıktılar: synapses: En yakın nodların doğru isimlerinin be-
        lirtildiği dosya

    '''

    import pandas as pd # Kütüphanelerin çalıştırılması

    reconst = pd.read_csv(reconst_file_name, sep = ' ', header =
0) # Dosyaların yüklenmesi
    synapses = pd.read_csv(synapse_file_name, header = 3)

    real_nodes = reconst['#id'][synapses['nearest_node']] # Doğru
en yakın nodun tespiti edilmesi
    synapses['real_nearest_node'] = real_nodes.reset_in-
dex()['#id']

    synapse_file_name_new = '%s_nodesfixed.csv' %(syn-
apse_file_name[:-4]) # Sinaps doyasının kaydedilmesi
    synapses.to_csv(synapse_file_name_new, index = False)

```

```

def get_real_nodes_to_isq_in_folder(filepath_from_main):
    '''
        Bu algoritma get_real_nodes_to_isq algoritmasını bir kortikal
        kolonda tespit edilen tüm sinaps parçaları için uygulamaktadır.
        Çalışması için get_real_nodes_to_isq algoritması aktive
        edilmelidir.

        Girdi: filepath_from_main: Ana dosyadan itibaren hedeflenen
        dosyaya uzanan hefef yolu
        Çıktı: Kortikal kolon içerisindeki Her bir sinaptik tespit
        bölgesi için doğru en yakın nodu içeren liste
    '''

```

```

'''

import os
import pandas as pd # Kütüphanelerin çalıştırılması

main_path = os.getcwd() # Dosya işlem merkezinin .tif ve .swc
uzantılı dosyaların bulunduğu bölgeye çekilmesi
file_path = os.path.join(main_path, filepath_from_main)
os.chdir(file_path)

folder_list = os.listdir()

for folder in folder_list:
    folder_path = os.path.join(file_path, folder)
    os.chdir(folder_path)

    file_list = os.listdir()

    for file in file_list: # Çalışma dosyası içinde dendritik
rekonstrüksiyon ve tespit edilen sinapsların dosyalarının tespit
edilmesi

        if '.swc' in file:
            reconst_file_name_ = file

        elif '.csv' in file:
            if '_nodesfixed' in file:
                continue
            else:
                synapse_file_name_ = file

        get_real_nodes_to_isq(reconst_file_name_ , syn-
apse_file_name_)

    os.chdir(main_path)

```

```

def fix_isq_synapse_loc(filepath_from_main, fixing_axis, ar-
ray_to_add):

```

```

'''

Dendritik rekonstrüksiyon ağacı sinapsların tespitinden önce
parçalandığı için
    tespit edilen sinapsların koordinatları asıl ağacın olduğu
bölgeye konumlandırılmalıdır.

```

Bunun sağlanması için dendritik rekonstrüksiyon ağacının kesimi sırasında elde edilen referans noktaları kullanılabilir.

Girdiler: filepath\_from\_main: Ana dosyadan itibaren hedeflenen dosyaya uzanan hedef yolu  
 fixing\_axis: Tespit edilen sinapsların düzeltilecek eksenini, kesim sırasında belirlenir  
 array\_to\_add: Sinapsın kortikal kolondaki esas koordinatı için eklenecek sayı, kesim sırasında belirlenir.  
 Veri tipi, uint64

Çıktılar: Kortikal kolon boyunca doğru sinaps koordinatlarını içeren .csv ve .marker uzantılı dosyalar.

```
'''
import os
import numpy as np
import pandas as pd # Kütüphanelerin çalıştırılması

main_path = os.getcwd() # Dosya işlem merkezinin .tif ve .swc
uzantılı dosyaların bulunduğu bölgeye çekilmesi
file_path = os.path.join(main_path, filepath_from_main)
os.chdir(file_path)

folder_list = os.listdir()
folder_array = np.array(folder_list, dtype = np.uint)

for folder in folder_list:
    num = np.array(folder, dtype = np.uint)
    folder_path = os.path.join(file_path, folder)
    os.chdir(folder_path)

    file_list = os.listdir()

    for file in file_list: # Tespit edilen sinaps dosya-
    larının okunması
        if 'nodesfixed.csv' in file:
            synapse_isq_file_name = file
        elif '.marker' in file:
            if '_coordfixed.marker' in file:
                continue
            else:
                synapse_marker_file_name = file

    synapse_isq = pd.read_csv(synapse_isq_file_name, header =
0)
    marker_isq = pd.read_csv(synapse_marker_file_name)
```

```

if fixing_axis == 'x':
    marker_fixing_axis = '##%s' %(fixing_axis)
else:
    marker_fixing_axis = fixing_axis

    synapse_isq[fixing_axis] = synapse_isq[fixing_axis] + array_to_add[num] # Tespit edilen sinaps koordinatlarının düzeltilmesi
    marker_isq[marker_fixing_axis] = marker_isq[marker_fixing_axis] + array_to_add[num]

    synapse_isq_file_name_new = '%s_coordfixed.csv' %(synapse_isq_file_name[:-4])
    synapse_marker_file_name_new = '%s_coordfixed.marker' %(synapse_marker_file_name[:-7])

    synapse_isq.to_csv(synapse_isq_file_name_new, index = False) # Dosyaların kaydedilmesi
    marker_isq.to_csv(synapse_marker_file_name_new, index = False)

os.chdir(main_path)

```

```

def merge_synapse_locs(filepath_from_main):

    '''
    Kortikal katmanlar boyunca her bir bölgede tespit edilen sinapsları birleştiren algoritma

    Girdi: filepath_from_main: Ana dosyadan itibaren hedeflenen dosyaya uzanan hedef yolu
    Çıktı: Tüm kortikal katmanlar boyunca tespit edilen sinapsların bulunduğu veriseti (tek renk)

    '''

    import os
    import numpy as np
    import pandas as pd # Kütüphanelerin çalıştırılması

    color_isq_full = []
    color_marker_full = []

```

```

    main_path = os.getcwd() # Dosya işlem merkezinin .tif ve .swc
uzantılı dosyaların bulunduğu bölgeye çekilmesi
    file_path = os.path.join(main_path, filepath_from_main)
    os.chdir(file_path)

    folder_list = os.listdir()
    for folder in folder_list:
        folder_path = os.path.join(file_path, folder)
        os.chdir(folder_path)

        file_list = os.listdir()

        for file in file_list: # Tespit edilen sinaps dosya-
ların okunması
            if 'coordfixed.csv' in file:
                synapse_isq_part_file_name = file
            elif 'coordfixed.marker' in file:
                synapse_marker_part_file_name = file

            synapse_isq_part = pd.read_csv(syn-
apse_isq_part_file_name, header = 0)
            synapse_marker_part = pd.read_csv(syn-
apse_marker_part_file_name)

            color_isq_full.append(synapse_isq_part)
            color_marker_full.append(synapse_marker_part)

            color_isq_full = pd.concat(color_isq_full) # Tespit edilen
sinaps koordinatlarının tüm kortikal katmanlar boyunca
birleştirilmesi
            color_isq_full['#id'] = np.arange(1,
color_isq_full.shape[0]+1)

            color_marker_full = pd.concat(color_marker_full)

            os.chdir(file_path)

            color_isq_full.to_csv('merged_isq.csv', index = False) #
Dosyaların kaydedilmesi
            color_marker_full.to_csv('merged.marker', index = False)

            os.chdir(main_path)

```

```

def merge_synapse_colors(filepath_from_main):
    """
    Bu algoritma VTA, VH ve BLA kaynaklı tespit edilmiş in-
    apsların birleştirilmesi için yazıldı.
    .csv uzantılı dosyalar basitçe birleştirilebilirken,
    .marker uzantılı dosyalar görsel fayda açısından
    renklendirilerek birleştirilmiştir.

    Girdi: filepath_from_main: Ana dosyadan itibaren hedeflenen
    dosyaya uzanan hedef yolu
    Çıktı: Tüm kortikal katmanlar boyunca tespit edilen sin-
    apsların bulunduğu veriseti (3 renk)
    """

    import os
    import numpy as np
    import pandas as pd # Kütüphanelerin çalıştırılması

    main_path = os.getcwd() # Dosya işlem merkezinin .tif ve .swc
    uzantılı dosyaların bulunduğu bölgeye çekilmesi
    file_path = os.path.join(main_path, filepath_from_main)
    os.chdir(file_path)

    file_list = os.listdir()

    for file in file_list: # Tespit edilen sinaps dosyalarının
    her bir renk için okunması
        if 'cyan' in file:
            if '.marker' in file:
                cyan_marker_file_name = file
            elif '.csv' in file:
                cyan_isq_file_name = file
        elif 'green' in file:
            if '.marker' in file:
                green_marker_file_name = file
            elif '.csv' in file:
                green_isq_file_name = file
        elif 'yellow' in file:
            if '.marker' in file:
                yellow_marker_file_name = file
            elif '.csv' in file:
                yellow_isq_file_name = file

```

```

cyan_marker = pd.read_csv(cyan_marker_file_name) # .marker
dosyalarındaki sinaps renklerinin esasına göre düzenlenmesi
cyan_marker[' color_r'] = 0
cyan_marker['color_g'] = 255
cyan_marker['color_b'] = 255

cyan_isq = pd.read_csv(cyan_isq_file_name, header = 0)
cyan_isq['color'] = 'cyan'

green_marker = pd.read_csv(green_marker_file_name)
green_marker[' color_r'] = 0
green_marker['color_g'] = 255
green_marker['color_b'] = 0

green_isq = pd.read_csv(green_isq_file_name, header = 0)
green_isq['color'] = 'green'

yellow_marker = pd.read_csv(yellow_marker_file_name)
yellow_marker[' color_r'] = 255
yellow_marker['color_g'] = 255
yellow_marker['color_b'] = 0

yellow_isq = pd.read_csv(yellow_isq_file_name, header = 0)
yellow_isq['color'] = 'yellow'

full_synapse_isq = pd.concat((cyan_isq, green_isq, yellow_isq)) # Tespit edilen sinapsların renk renk birleştirilmesi
full_synapse_isq['#id'] = np.arange(1, full_synapse_isq.shape[0]+1)

full_synapse_marker = pd.concat((cyan_marker, green_marker, yellow_marker))

full_synapse_isq.to_csv('all_colors_isq_merged.csv', index = False) # Dosyaların kaydedilmesi
full_synapse_marker.to_csv('all_colors_merged.marker', index = False)

os.chdir(main_path)

```

```

def add_tree_segment(filepath_from_main):
    """
    Bu algoritma tespit edilen orijinal en yakın nodun bulunduğu
    dendritik segment ve dendritik ağaç dalının etiketini ekler.
    Bu bilgi Gliko ve arkadaşlarının rekonstrüksiyon algorit-
    masının kullanımı sonrasında elde edilir.

    Girdi: filepath_from_main: Ana dosyadan itibaren hedeflenen
    dosyaya uzanan hedef yolu
    """

    import os
    import numpy as np
    import pandas as pd # Kütüphanelerin çalıştırılması

    main_path = os.getcwd() # Dosya işlem merkezinin .tif ve .swc
    uzantılı dosyaların bulunduğu bölgeye çekilmesi
    file_path = os.path.join(main_path, filepath_from_main)
    os.chdir(file_path)

    file_list = os.listdir()

    for file in file_list:
        if '.csv' in file:
            if 'all' in file:
                isq_file_name = file
            else:
                continue
            elif 'swc' in file:
                reconst_file_name = file
            else:
                continue

    synapses = pd.read_csv(isq_file_name, header = 0) # Sinaps ve
    dendritik rekonstrüksiyon dosyalarının okunması
    reconst = pd.read_csv(reconst_file_name, sep = ' ', header =
0)

    trees = []
    segments = []

    for i in np.arange(len(synapses)):
        node = synapses['real_nearest_node'][i]
        loc = np.where(node == reconst['#id'])

        tree = reconst['tree'].iloc[loc]
        segment = reconst['segment'].iloc[loc]

```



```

trees = np.append(trees, tree)
segments = np.append(segments, segment)

synapses['tree'] = trees.astype(np.uint) # Dendritik segment
ve ağaç bilgilerinin eklenmesi
synapses['segment'] = segments.astype(np.uint)

synapses.to_csv('%s_treesadded.csv' %(isq_file_name[:-4]),
index = False) # Tespit edilen sinaps verisinin kaydedilmesi

os.chdir(main_path)

```

```

def add_layers(filepath_from_main, check_index, L1_limit,
L2_3_limit, L5_limit, L6_limit):

    '''
    Bu algoritma dendritik rekonstrüksiyon ağacı ve üzerindeki
    tespit edilen sinapsların hangi kortikal katmana ait olduğunu
    etiketler.
    Kortikal katman sınırları korteksin boyutlarına göre kortikal
    kolon boyunca sahip oldukları alana göre tespit edilir.

    Girdiler: filepath_from_main: Ana dosyadan itibaren
    hedeflenen dosyaya uzanan hedef yolu
             check_index: Kortikal katmanların uzandığı
    eksen
             L1_limit: 1. katmanın sınırları
             L2_3_limit: 2/3. katmanın sınırları
             L5_limit: 5. katmanın sınırları
             L6_limit: 6. katmanın sınırları

    Çıktılar: synapses: Tespit edilen sinapsları içeren veri
    seti: Bu veri setinde sinapsın en yakın olduğu nod,
    üzerinde bulunduğu ağaç ve segment numarası,
    bulunduğu kortikal katman ve hangi beyin bölgesinden
    kaynaklandığı bilgilerini içerir.
             reconst: Katman bilgisini içeren dendritik
    rekonstrüksiyon ağacı
    '''

    import os
    import numpy as np

```

```

import pandas as pd # Kütüphanelerin çalıştırılması

main_path = os.getcwd() # Dosya işlem merkezinin .tif ve .swc
uzantılı dosyaların bulunduğu bölgeye çekilmesi
file_path = os.path.join(main_path, filepath_from_main)
os.chdir(file_path)

file_list = os.listdir()

for file in file_list:
    if '.csv' in file:
        if 'treesadded' in file:
            if 'layersadded' in file:
                continue
            else:
                isq_file_name = file
        else:
            continue
    elif '.swc' in file:
        if 'dendrite_' in file:
            continue
        else:
            reconst_file_name = file
    else:
        continue

reconst = pd.read_csv(reconst_file_name, header = 0, sep = '
') # Sinaps ve dendritik rekonstrüksiyon dosyalarının okunması
print(reconst_file_name)
synapses = pd.read_csv(isq_file_name, header = 0)
synapses['layer'] = 'unknown'
reconst['layer'] = 'unknown'

if L1_limit[0] > L1_limit[1]: # katman koordinatlarının
düzenlenmesi
    L1_limit = np.flip(L1_limit)

if L2_3_limit[0] > L2_3_limit[1]:
    L2_3_limit = np.flip(L2_3_limit)

if L5_limit[0] > L5_limit[1]:
    L5_limit = np.flip(L5_limit)

if L6_limit[0] > L6_limit[1]:
    L6_limit = np.flip(L6_limit)

```

```

layer_locs = [L1_limit, L2_3_limit, L5_limit, L6_limit]
upper_most = max(max(layer_locs))
lower_most = min(min(layer_locs))

if upper_most > lower_most:
    surface_deep = [lower_most, upper_most]
else:
    surface_deep = [upper_most, lower_most]

L1_place = np.where((synapses[check_index] >= L1_limit[0]) &
(synapses[check_index] < L1_limit[1]))
L2_3_place = np.where((synapses[check_index] >=
L2_3_limit[0]) & (synapses[check_index] < L2_3_limit[1]))
L5_place = np.where((synapses[check_index] >= L5_limit[0]) &
(synapses[check_index] < L5_limit[1]))
L6_place = np.where((synapses[check_index] >= L6_limit[0]) &
(synapses[check_index] < L6_limit[1]))

synapses['layer'].iloc[L1_place] = 'L1' # Sinapsların ve
nodların bulunduğu katmanların tanımlanması
synapses['layer'].iloc[L2_3_place] = 'L2/3'
synapses['layer'].iloc[L5_place] = 'L5'
synapses['layer'].iloc[L6_place] = 'L6'

real_layer_places = np.where((reconst[check_index] > sur-
face_deep[0]) & (reconst[check_index] < surface_deep[1]))
reconst = reconst.iloc[real_layer_places]

L1_place = np.where((reconst[check_index] >= L1_limit[0]) &
(reconst[check_index] < L1_limit[1]))
L2_3_place = np.where((reconst[check_index] >= L2_3_limit[0])
& (reconst[check_index] < L2_3_limit[1]))
L5_place = np.where((reconst[check_index] >= L5_limit[0]) &
(reconst[check_index] < L5_limit[1]))
L6_place = np.where((reconst[check_index] >= L6_limit[0]) &
(reconst[check_index] < L6_limit[1]))

reconst['layer'].iloc[L1_place] = 'L1'
reconst['layer'].iloc[L2_3_place] = 'L2/3'
reconst['layer'].iloc[L5_place] = 'L5'
reconst['layer'].iloc[L6_place] = 'L6'

synapses.to_csv('%s_layersadded.csv' %(isq_file_name[:-4]),
index = False) # Dosyaların kaydedilmesi

```

```
reconst.to_csv('%s_layersfixed.swc' %(reconst_file_name[:-4]), index = False, sep = ' ')

os.chdir(main_path)
```

## EK-2: Veri Analizi Sırasında Kullanılan Algoritmalar

```
def get_branch_lengths(reconst, px_size, show_plot = False):
    """
    Bu algoritma dendrit rekonstrüksiyon ağacındaki her bir dendritik segmentin µm cinsinden uzunluğunu hesaplar.

    Girdiler:
    reconst: .swc uzantılı dendritik rekonstrüksiyon dosyası
    px_size: Veri kazanımı sırasında konfokal mikroskopide düzenlenmiş xy piksel boyutu. Veri tipi: float32
    show_plot: Ölçülen segment uzunluklarının gösterilmesini sağlayan yöneltici. Veri tipi: binary

    Çıktılar:
    segment_lengths: Dendrit rekonstrüksiyon ağacındaki her bir segmentin µm uzunluğu. Veri tipi: float32, array
    segments: Dendrit rekonstrüksiyon ağacındaki tüm segmentler. Veri tipi: uint32
    Segment uzunluk histogramı show_plot = True ise gösterir.

    """

    import numpy as np      # Kütüphanelerin yüklenmesi
    import pandas as pd
    import matplotlib.pyplot as plt

    segment_lengths = []
    segments = []

    if len(reconst) == 0:
        return segment_lengths, segments

    segments = np.unique(reconst['segment'])

    for segment in segments: # Segment uzunluklarının hesaplanıp dizilmesi

        nodes_in_branch = np.where(reconst['segment'] == segment)
        segment_index = reconst.iloc[nodes_in_branch].index
        segment_length = 0
        for node in np.arange(len(segment_index)-1):
            pre_node = segment_index[node]
            post_node = segment_index[node+1]
```

```

x_pre = reconst['x'][pre_node]
y_pre = reconst['y'][pre_node]
z_pre = reconst['z'][pre_node]

x_post = reconst['x'][post_node]
y_post = reconst['y'][post_node]
z_post = reconst['z'][post_node]

dx = x_post - x_pre
dy = y_post - y_pre
dz = z_post - z_pre

delta = np.sqrt(dx**2 + dy**2 + dz**2)
segment_length += delta

segment_lengths = np.append(segment_lengths, segment_length)

segment_lengths = segment_lengths * px_size

if show_plot == True: # Segment uzunluk histogramının
çıkarılması
    segment_start = 0
    segment_step = 1
    segment_stop = np.max(segment_lengths) + branch_step
    segment_bins = np.arange(segment_start, segment_stop,
segment_step)

    plt.hist(segment_lengths, bins = segment_bins)
    plt.xlabel('Segment lengths (um)')
    plt.ylabel('Frequency')

return segment_lengths, segments

```

```
def get_connected_dendrite_numbers(filepath_from_main):
```

```
    ...
```

Bu algoritma bütün kesitlerdeki sinaps içeren ve içermeyen dendritik segmentleri tespit eder. Her birisinin uzunluğunu ölçer.

Her bir segment tipinin sayısını hesaplar. Segmentlerdeki sinaps sayısını tespit eder.

Girdi:

filepath\_from\_main: Ana dosyadan itibaren hedeflenen dosyaya uzanan hedef yolu. Bu yolun sonunda tüm kesitlerin sinaps dendritik veri setleri son haline getirilmiş olmalıdır.

Algoritma işlem klasörünü kesit kesit işleyecek şekilde değişebilir.

Çıktı:

Sinaptik bağlantı veri seti: .pkl uzantılıdır. Çıkan veriseti mevcut pandas algoritmalarıyla okunabilmektedir.

İçerisinde kesitin hangi fareye ait olduğu, hangi segmentleri içerdiği, bu segmentlerden ne kadar bulunduğu bilgisi mevcuttur.

```
'''

import os
import numpy as np      # Kütüphanelerin yüklenmesi
import pandas as pd

main_path = os.getcwd()
folder_path = os.path.join(main_path, filepath_from_main)
os.chdir(folder_path)

folder_list = os.listdir()

total_num = 0

dframe = {'mouse': [], # Boş bağlantı verisetinin oluşturulması
          'total_num_segments': [],
          'segments_with_no_synapses': [],

          'cyan_only_branch_number': [],
          'cyan_only_segments': [],
          'syn_num_cyan_only_segments': [],
          'total_syn_num_cyan_only_segments': [],
          'cyan_only_branch_lengths': [],

          'green_only_branch_number': [],
          'green_only_segments': [],
          'syn_num_green_only_segments': [],
          'total_syn_num_green_only_segments': [],
          'green_only_branch_lengths': [],

          'yellow_only_branch_number': [],
          'yellow_only_segments': [],
          'syn_num_yellow_only_segments': [],
```

```

'total_syn_num_yellow_only_segments': [],
'yellow_only_branch_lengths': [],

'cyan_green_not_yellow_branch_number': [],
'cyan_green_not_yellow_segments': [],
'syn_num_cyan_green_not_yellow_segments': [],
'total_syn_num_cyan_green_not_yellow_segments': [],
'cyan_green_not_yellow_branch_lengths': [],

'cyan_yellow_not_green_branch_number': [],
'cyan_yellow_not_green_segments': [],
'syn_num_cyan_yellow_not_green_segments': [],
'total_syn_num_cyan_yellow_not_green_segments': [],
'cyan_yellow_not_green_branch_lengths': [],

'green_yellow_not_cyan_branch_number': [],
'green_yellow_not_cyan_segments': [],
'syn_num_green_yellow_not_cyan_segments': [],
'total_syn_num_green_yellow_not_cyan_segments': [],
'green_yellow_not_cyan_branch_lengths': [],

'cyan_green_yellow_branch_number': [],
'cyan_green_yellow_segments': [],
'syn_num_cyan_green_yellow_segments': [],
'total_syn_num_cyan_green_yellow_segments': [],
'cyan_green_yellow_branch_lengths': []}

for folder in folder_list:
    if '.' in folder:
        continue

    file_path = os.path.join(folder_path, folder)
    os.chdir(file_path)

    file_list = os.listdir()

    for file in file_list: # Tepit edilen sinapsların ve den-
        dritik rekonstrüksiyon ağacının yüklenmesi
            if '.csv' in file:
                if 'filtered' in file:
                    synapse_file_name = file
                else:
                    continue
            if '.swc' in file:
                if 'filtered.swc' in file:
                    reconst_file_name = file
                else:

```



```

        continue

    else:
        continue

    synapses = pd.read_csv(synapse_file_name, header = 0)
    dendrites = pd.read_csv(reconst_file_name, header = 0,
sep = ' ')

    segment_lengths, segments = get_branch_lengths(dendrites,
px_size = 0.103)
    non_zero_where = np.where(segment_lengths != 0)
    non_zero_segments = segments[non_zero_where]

    number_of_allsegments = len(non_zero_segments)

    connected_segments = np.unique(synapses['segment'])

    cyan = 0
    cyan_B = []
    cyan_BL = []

    green = 0
    green_BL = []
    green_B = []

    yellow = 0
    yellow_B = []
    yellow_BL = []

    cyan_green = 0
    cyan_green_B = []
    cyan_green_BL = []

    cyan_yellow = 0
    cyan_yellow_B = []
    cyan_yellow_BL = []

    green_yellow = 0
    green_yellow_B = []
    green_yellow_BL = []

    cyan_green_yellow = 0
    cyan_green_yellow_B = []
    cyan_green_yellow_BL = []

```

```

        for segment in connected_segments: # Her bir segmentin
uzunluğunun ölçülmesi bu segmentlerdeki mevcut sinapsların tespit
edilmesi, gruplanması ve sayılması
            segments_where = np.where(synapses['segment'] == seg-
ment)
            branch_where = np.where(dendrites['segment'] == seg-
ment)

            connected_synapses_on_segment = synapses.iloc[seg-
ments_where]
            colors_in_connected_synapses = np.unique(con-
nected_synapses_on_segment['color'])
            the_branch = dendrites.iloc[branch_where]
            branch_length, branch =
get_branch_lengths(the_branch, px_size = 0.103)

            if branch_length == 0:
                continue

            if 'cyan' in colors_in_connected_synapses:
                cyan += 1
                cyan_B = np.append(cyan_B, branch)

            if 'green' in colors_in_connected_synapses:
                green += 1
                green_B = np.append(green_B, branch)

            if 'yellow' in colors_in_connected_synapses:
                yellow += 1
                yellow_B = np.append(yellow_B, branch)

            if 'cyan' in colors_in_connected_synapses and 'green'
in colors_in_connected_synapses:
                cyan_green += 1
                cyan_green_B = np.append(cyan_green_B, branch)

            if 'cyan' in colors_in_connected_synapses and 'yel-
low' in colors_in_connected_synapses:
                cyan_yellow += 1
                cyan_yellow_B = np.append(cyan_yellow_B, branch)

            if 'green' in colors_in_connected_synapses and 'yel-
low' in colors_in_connected_synapses:
                green_yellow += 1

```

```

        green_yellow_B = np.append(green_yellow_B,
branch)

        if 'cyan' in colors_in_connected_synapses and 'green'
in colors_in_connected_synapses and 'yellow' in colors_in_con-
nected_synapses:
            cyan_green_yellow += 1
            cyan_green_yellow_B = np.append(cyan_green_yel-
low_B, branch)

        '''
        Sinaps bulunduran dendritik segmentlerin ve uzun-
luklarının hesaplanması
        '''
        cyan_green_mutual = np.isin(cyan_green_B, cyan_green_yel-
low_B)
        cyan_green_B = cyan_green_B[np.invert(cyan_green_mutual)]
        cyan_green_SN = get_syn_numbers_on_segments(cyan_green_B,
synapses)
        cyan_green_BL = segment_lengths[np.isin(segments,
cyan_green_B)]

        cyan_yellow_mutual = np.isin(cyan_yellow_B,
cyan_green_yellow_B)
        cyan_yellow_B = cyan_yellow_B[np.invert(cyan_yellow_mu-
tual)]
        cyan_yellow_SN = get_syn_numbers_on_segments(cyan_yel-
low_B, synapses)
        cyan_yellow_BL = segment_lengths[np.isin(segments,
cyan_yellow_B)]

        green_yellow_mutual = np.isin(green_yellow_B,
cyan_green_yellow_B)
        green_yellow_B = green_yellow_B[np.invert(green_yel-
low_mutual)]
        green_yellow_SN = get_syn_numbers_on_segments(green_yel-
low_B, synapses)
        green_yellow_BL = segment_lengths[np.isin(segments,
green_yellow_B)]

        mutual = np.hstack([cyan_green_B, cyan_yellow_B,
cyan_green_yellow_B])
        cyan_mutual = np.isin(cyan_B, mutual)
        cyan_B = cyan_B[np.invert(cyan_mutual)]
        cyan_SN = get_syn_numbers_on_segments(cyan_B, synapses)

```

```

cyan_BL = segment_lengths[np.isin(segments, cyan_B)]

mutual = np.hstack([cyan_green_B, green_yellow_B,
cyan_green_yellow_B])
green_mutual = np.isin(green_B, mutual)
green_B = green_B[np.invert(green_mutual)]
green_SN = get_syn_numbers_on_segments(green_B, synapses)
green_BL = segment_lengths[np.isin(segments, green_B)]

mutual = np.hstack([green_yellow_B, cyan_yellow_B,
cyan_green_yellow_B])
yellow_mutual = np.isin(yellow_B, mutual)
yellow_B = yellow_B[np.invert(yellow_mutual)]
yellow_SN = get_syn_numbers_on_segments(yellow_B, synap-
ses)
yellow_BL = segment_lengths[np.isin(segments, yellow_B)]

cyan_green_yellow_SN = get_syn_numbers_on_seg-
ments(cyan_green_yellow_B, synapses)
cyan_green_yellow_BL = segment_lengths[np.isin(segments,
cyan_green_yellow_B)]

'''
Sinaps içeren segmentsayılarının tespit edilmesi
'''

cyan_green_no_yellow = cyan_green - cyan_green_yellow
cyan_yellow_no_green = cyan_yellow - cyan_green_yellow
green_yellow_no_cyan = green_yellow - cyan_green_yellow

only_cyan = cyan - cyan_green_no_yellow - cyan_yel-
low_no_green - cyan_green_yellow
only_green = green - cyan_green_no_yellow - green_yel-
low_no_cyan - cyan_green_yellow
only_yellow = yellow - cyan_yellow_no_green - green_yel-
low_no_cyan - cyan_green_yellow

only_one_synapse = only_cyan + only_green + only_yellow
two_synapses = cyan_green_no_yellow + cyan_yel-
low_no_green + green_yellow_no_cyan
num_connected_segments = only_one_synapse + two_synapses
+ cyan_green_yellow

num_unconn_segments = number_of_allsegments - num_con-
nected_segments

'''

```

```

Veri tabanı oluşturma
'''

print('Mouse: ', folder)
dframe['mouse'].append(folder)
dframe['total_num_segments'].append(number_of_allseg-
ments)
dframe['segments_with_no_synapses'].append(num_un-
conn_segments)

dframe['cyan_only_branch_number'].append(only_cyan)
dframe['cyan_only_segments'].append(cyan_B)
dframe['syn_num_cyan_only_segments'].append(cyan_SN)
dframe['total_syn_num_cyan_only_segments'].ap-
pend(np.sum(cyan_SN))
dframe['cyan_only_branch_lengths'].append(np.ar-
ray(cyan_BL))

dframe['green_only_branch_number'].append(only_green)
dframe['green_only_segments'].append(green_B)
dframe['syn_num_green_only_segments'].append(green_SN)
dframe['total_syn_num_green_only_segments'].ap-
pend(np.sum(green_SN))
dframe['green_only_branch_lengths'].append(np.ar-
ray(green_BL))

dframe['yellow_only_branch_number'].append(only_yellow)
dframe['yellow_only_segments'].append(yellow_B)
dframe['syn_num_yellow_only_segments'].append(yellow_SN)
dframe['total_syn_num_yellow_only_segments'].ap-
pend(np.sum(yellow_SN))
dframe['yellow_only_branch_lengths'].append(np.array(yel-
low_BL))

dframe['cyan_green_not_yellow_branch_number'].ap-
pend(cyan_green_no_yellow)
dframe['cyan_green_not_yellow_segments'].ap-
pend(cyan_green_B)
dframe['syn_num_cyan_green_not_yellow_segments'].ap-
pend(cyan_green_SN)
dframe['total_syn_num_cyan_green_not_yellow_seg-
ments'].append(np.sum(cyan_green_SN))
dframe['cyan_green_not_yellow_branch_lengths'].ap-
pend(np.array(cyan_green_BL))

```

```

        dframe['cyan_yellow_not_green_branch_number'].append(cyan_yellow_no_green)
        dframe['cyan_yellow_not_green_segments'].append(cyan_yellow_B)
        dframe['syn_num_cyan_yellow_not_green_segments'].append(cyan_yellow_SN)
        dframe['total_syn_num_cyan_yellow_not_green_segments'].append(np.sum(cyan_yellow_SN))
        dframe['cyan_yellow_not_green_branch_lengths'].append(np.array(cyan_yellow_BL))

        dframe['green_yellow_not_cyan_branch_number'].append(green_yellow_no_cyan)
        dframe['green_yellow_not_cyan_segments'].append(green_yellow_B)
        dframe['syn_num_green_yellow_not_cyan_segments'].append(green_yellow_SN)
        dframe['total_syn_num_green_yellow_not_cyan_segments'].append(np.sum(green_yellow_SN))
        dframe['green_yellow_not_cyan_branch_lengths'].append(np.array(green_yellow_BL))

        dframe['cyan_green_yellow_branch_number'].append(cyan_green_yellow)
        dframe['cyan_green_yellow_segments'].append(cyan_green_yellow_B)
        dframe['syn_num_cyan_green_yellow_segments'].append(cyan_green_yellow_SN)
        dframe['total_syn_num_cyan_green_yellow_segments'].append(np.sum(cyan_green_yellow_SN))
        dframe['cyan_green_yellow_branch_lengths'].append(np.array(cyan_green_yellow_BL))

    '''
    print('Number of all segments:', number_of_allsegments) # İsteyen bir kişi burayı açarak veri setindeki sinaps sayılarını inceleyebilir.
    print('Number of unconnected segments:', num_unconn_segments)

    print('Number of cyan only segments:', only_cyan)
    print('Number of green only segments:', only_green)
    print('Number of yellow only segments:', only_yellow)
    print('Number of cyan and green connected segments: ', cyan_green_no_yellow)

```

```

        print('Number of cyan and yellow connected segments: ',
cyan_yellow_no_green)
        print('Number of green and yellow connected segments: ',
green_yellow_no_cyan)
        print('Number of segments with all synapses connected:',
cyan_green_yellow)
'''
dframe = pd.DataFrame(data = dframe)
os.chdir(main_path)
dframe.to_pickle('connection_data.pkl') # Veri setinin
kaydedilmesi

```

```

def draw_synaptic_density_volume(filepath_from_main,
                                mice_data_file_name,
                                check_axis,
                                mode = 'equalize',
                                check_points = [0,1],
                                num_of_steps = 20,
                                px_size = 0.103,
                                show_plots = False):

'''
    Bu algoritma bir beyin kesitinde katmanlar boyunca toplam den-
dritik segment uzunluğunu, VTA, VH ve BLA kaynaklı sinapsların
sayısını,
    hacimsel yoğunluğunu, dendritik uzunluğa düşen sinaps sayısını
hesaplar.
    Bu algoritma çalışma sırasında kiremit tarama yönünü önemsemez.
Bu sebeple kullanıcının işlem yönünü belirtmesi gereklidir.
    Bu algoritma için kullanılacak sinaptik veri tabanı ve den-
dritik rekonstrüksiyon görüntüsünün hatalı analiz sonucu eldes-
inin önlenmesi için kenarlarından kırılması tavsiye
edilir.(...filtered.swc ve ...filtered.csv)
    Dendritik uzunluğa düşen sinaptik yoğunluk analizlerinde katman
katman ayrılmış dendritik rekonstrüksiyonların kullanılması
tavsiye edilir. Bu özellikle verilerin eşit parçalara ayrıldığı
seçenekte etkili olabilir.

    Girdiler:
    filepath_from_main: Ana dosyadan itibaren hedeflenen dosyaya
uzanan hedef yolu.
    mice_data_file_name: .csv uzantılıdır. Elle hazırlanmış bu
dosya kesitlerin hangi fareye ait olduğunu, farelerin davranış fe-
notiplerini, kesit kalınlıklarını ve eğer içermiyorsa 1. katman-
dan ne kadar kaybettiğinin bilgisini içermelidir.
    check_axis: İşlem hesap yönü. 'x' veya 'y' olabilir.

```

`mode`: Kortikal görüntünün nasıl parçalanacağını ifade eder. 'equalize' olursa veri eşit parçalara ayrılır böylece bir ekseninde yoğunluk incelemesi için fırsat bulunabilir. 'layers' olursa katman oran sınırlarının tanımlanması gereklidir.

`check_points`: Katman sınır oranları. Liste şeklinde dizilir. `mode`, 'layers' olursa kortikal katmanların çizgisel oranları yazılmalıdır. Ör: [0,0.2,0.6,1]

`num_of_steps`: Kortikal verinin ayrılacağı eşit sayıda parça sayısı. `mode`, 'equalize' olursa kullanılır.

`px_size`: Görüntü alımı sırasında ayarlanan xy piksel mesafesi

`show_plots`: Yoğunluk, sayım verilerinin incelenen eksene göre grafiğini çıkaran emir. Veri tipi: binary

Çıktı:

Veriseti: .pkl uzantılı veri seti. İncelenen kortikal görüntüye ait fare, her bir katmanın piksel ve  $\mu\text{m}$  cinsinden sınırları, her katmandaki VTA, VH ve BLA kaynaklı sinaps sayıları ve dendiritik uzunluğa düşen sinaps yoğunluğunu içerir.

'''

```
import os
import numpy as np      # Kütüphanelerin yüklenmesi
import pandas as pd
import matplotlib.pyplot as plt

mice_data = pd.read_csv('%s.csv' %(mice_data_file_name),
header = 0)

main_path = os.getcwd()
folder_path = os.path.join(main_path, filepath_from_main)
os.chdir(folder_path)

folder_list = os.listdir()

dframe = {'mouse': [], # Verisetinin kurulması
          'depth_in_um' : [],
          'total_dend_in_um': [],
          'check_px': [],
          'cyan_number': [],
          'green_number': [],
          'yellow_number': [],
          'cyan_dens': [],
          'green_dens': [],
          'yellow_dens': []}

if mode == 'equalize': # İnceleme modunun belirlenmesi
```



```

    check_points = np.linspace(0,1,num_of_steps)
elif mode == 'layers':
    check_points = np.array(check_points)
    num_of_steps = len(check_points)

    for folder in folder_list: # Dendiritik rekonstrüksiyon ve
sinaps verisinin yüklenmesi
        if '.' in folder:
            continue

    file_path = os.path.join(folder_path, folder)
    os.chdir(file_path)

    file_list = os.listdir()

    for file in file_list:
        if '.csv' in file:
            if 'filtered' in file:
                synapse_file_name = file
            else:
                continue
        if '.swc' in file:
            if 'filtered.swc' in file:
                reconst_file_name = file
            elif 'L1' in file:
                L1_reconst_file_name = file
            elif 'L2-3' in file:
                L23_reconst_file_name = file
            elif 'L5' in file:
                L5_reconst_file_name = file
            elif 'L6' in file:
                L6_reconst_file_name = file
            else:
                continue
        else:
            continue

    synapses = pd.read_csv(synapse_file_name, header = 0)
    cur_reconst = pd.read_csv(reconst_file_name, header = 0,
sep = ' ')

    linked_branches = np.unique(synapses['segment'])
    all_branches = np.unique(cur_reconst['segment'])
    linked_synapses =
linked_branches[np.isin(linked_branches, all_branches)]
    synapses = synapses.iloc[np.isin(synapses['seg-
ment'],linked_synapses)]

```

```

        blengths, _ = get_branch_lengths(cur_reconst, px_size =
px_size)

        total_branch_length_incolumn = np.sum(blengths)

        L1_reconst = pd.read_csv(L1_reconst_file_name, header =
0, sep = ' ') # Her bir katmandaki dendritik segment uzunluğunun
hesaplanması
        L1_reconst_branch_lengths, _ = get_branch_lengths(L1_re-
const, px_size = px_size)
        L1_total_branch_length = np.sum(L1_re-
const_branch_lengths)

        L23_reconst = pd.read_csv(L23_reconst_file_name, header =
0, sep = ' ')
        L23_reconst_branch_lengths, _ =
get_branch_lengths(L23_reconst, px_size = px_size)
        L23_total_branch_length = np.sum(L23_re-
const_branch_lengths)

        L5_reconst = pd.read_csv(L5_reconst_file_name, header =
0, sep = ' ')
        L5_reconst_branch_lengths, _ = get_branch_lengths(L5_re-
const, px_size = px_size)
        L5_total_branch_length = np.sum(L5_re-
const_branch_lengths)

        L6_reconst = pd.read_csv(L6_reconst_file_name, header =
0, sep = ' ')
        L6_reconst_branch_lengths, _ = get_branch_lengths(L6_re-
const, px_size = px_size)
        L6_total_branch_length = np.sum(L6_re-
const_branch_lengths)

        if max(synapses['y']) > max(synapses['x']): # İnceleme
yönünün belirlenmesi
            layer_axis = 'y'
        else:
            layer_axis = 'x'

        if check_axis == 'y':
            other_axis = 'x'
        else:

```

```

        other_axis = 'y'

        find_mouse = np.where(mice_data['mouse'] == folder)
        d_L1 = mice_data['L1_from_um'].iloc[find_mouse]
        d_cortex = mice_data['cortex_depth_in_um'].iloc[find_mouse]
        d_pixel = mice_data['cortex_img_depth_px'].iloc[find_mouse]

        if L1_total_branch_length == 0: # Eğer yoksa L1'in
alınmadığı durumda tespit edilen olası kaybın işleme alınması
(Kayıp reflektans görüntü incelemesi esnasında tespit edilir.)

            add_to_col = d_pixel * (d_L1/d_cortex)
            add_to_col = np.array(add_to_col)[0]
            synapses[layer_axis] = synapses[layer_axis] +
add_to_col
            cur_reconst[layer_axis] = cur_reconst[layer_axis] +
add_to_col

        colors = np.unique(synapses['color'])
        layers = np.unique(synapses['layer'])

        print('Mouse: ', folder)
        dframe['mouse'].append(folder)

        if max(synapses[check_axis]) > max(synapses[other_axis]):
            if L1_total_branch_length == 0:
                max_of_check_axis = d_pixel * (1 + (d_L1/d_cortex))
            else:
                max_of_check_axis = d_pixel

        check = np.array(max_of_check_axis)[0] * check_points
        else:
            max_of_check_axis = max(cur_reconst[check_axis])
            check = max_of_check_axis * check_points

        dframe['check_px'].append(check)
        check_p = []

        depth_of_reconst = max(cur_reconst['z']) - min(cur_reconst['z'])

        total_branch_lengths = []

```

```

        for c in np.arange(num_of_steps-1): # Her bir inceleme
            odağındaki dendritik segment uzunluğu, sinaps sayısının tespiti
            (katman veya eşit parça)
                check_p = np.append(check_p, (check[c]+check[c+1])/2)
                dend_between_axis_values = np.where((cur_re-
const[check_axis] > check[c]) &
                                                    (cur_re-
const[check_axis] < check[c+1]))
                reconst_between_axis_values = cur_re-
const.iloc[dend_between_axis_values]
                branch_lengths_between_axis_values, _ =
get_branch_lengths(reconst_between_axis_values,

                px_size = px_size)
                total_branch_length_between_axis_values =
np.sum(branch_lengths_between_axis_values)
                total_branch_lengths = np.append(to-
tal_branch_lengths, total_branch_length_between_axis_values)

                plt.figure()
                plt.plot(check_p*px_size, total_branch_lengths)
                plt.xlabel('Distance from surface (um)')
                plt.ylabel('Dendrite Length (um)')
                plt.title('Dendrite lengths of mice: %s' %(folder))

                if check_axis == layer_axis:
                    for layer in layers:
                        layer_place = np.where(synapses['layer'] ==
layer)

                        axis_locs = synap-
ses[check_axis].iloc[layer_place]
                        max_layer = max(axis_locs) * px_size
                        plt.axvline(x = max_layer, linestyle = '--',
color = 'k')

                plt.figure(figsize = (4.8,33.8))

                depth_in_um = check_p*px_size
                dframe['depth_in_um'].append(np.array(depth_in_um))
                dframe['total_dend_in_um'].append(total_branch_lengths)

                for color in colors:

                    color_syn_place = np.where(synapses['color'] ==
color)

                    synapses_color = synapses.iloc[color_syn_place]
                    num_syn_color_incolumn = len(synapses_color)

```

```

        nums_along_axis = []
        denses_along_axis = []
        syn_divby_dend_lenght_along_axis = []
        syn_divby_dend_lenght_of_dominant_layer_along_axis =
[]
        syn_divby_total_syn_incolumn_array = []
        syn_divby_total_syn_incolumn__divby__lenght_be-
tween_axis_values_divby_total_dend_lenght_array = []

        for c in np.arange(1, num_of_steps):

            syn_between_axis_values = np.where((synap-
ses_color[check_axis] > check[c-1]) &
                                                (synap-
ses_color[check_axis] < check[c]))

            dend_between_axis_values = np.where((cur_re-
const[check_axis] > check[c-1]) &
                                                (cur_re-
const[check_axis] < check[c]))

            reconst_between_axis_values = cur_re-
const.iloc[dend_between_axis_values]

            width_of_other_axis = get_width_of_reconst(re-
const_between_axis_values, check_axis) * px_size

            if len(reconst_between_axis_values) == 0:
                dominant_layer = 'L1'
            else:
                dominant_layer = reconst_between_axis_val-
ues['layer'].value_counts().index[0]

            if dominant_layer == 'L1':
                use_for_div = L1_total_branch_length
            elif dominant_layer == 'L2/3':
                use_for_div = L23_total_branch_length
            elif dominant_layer == 'L5':
                use_for_div = L5_total_branch_length
            elif dominant_layer == 'L6':
                use_for_div = L6_total_branch_length

            num_between_axis_values = len(syn_be-
tween_axis_values[0])

```

```

        dens_between_axis_values = num_between_axis_val-
ues / (depth_of_reconst * width_of_other_axis * (check[c] -
check[c-1]) * px_size)
        syn_divby_dend_lenght_between_axis_values =
num_between_axis_values / total_branch_lengths[c-1]
        syn_divby_dend_lenght_of_dominant_layer = num_be-
tween_axis_values / use_for_div
        syn_divby_total_syn_incolumn = num_be-
tween_axis_values/num_syn_color_incolumn
        syn_divby_total_syn_incolumn__divby__lenght_be-
tween_axis_values_divby_total_dend_lenght = syn_divby_to-
tal_syn_incolumn/(total_branch_lengths[c-1]/to-
tal_branch_lenght_incolumn)

        nums_along_axis = np.append(nums_along_axis,
num_between_axis_values)
        denses_along_axis = np.append(denses_along_axis,
dens_between_axis_values)
        syn_divby_dend_lenght_along_axis = np.ap-
pend(syn_divby_dend_lenght_along_axis,


syn_


divby_dend_lenght_between_axis_values)
        syn_divby_dend_lenght_of_domi-
nant_layer_along_axis = np.append(syn_divby_dend_lenght_of_domi-
nant_layer_along_axis,


syn_


syn_divby_dend_lenght_of_dominant_layer)
        syn_divby_total_syn_incolumn_array = np.ap-
pend(syn_divby_total_syn_incolumn_array,


sy


n_divby_total_syn_incolumn)
        syn_divby_total_syn_incolumn__divby__lenght_be-
tween_axis_values_divby_total_dend_lenght_array = np.ap-
pend(syn_divby_total_syn_incolumn__divby__lenght_be-
tween_axis_values_divby_total_dend_lenght_array,


syn_divby_


total_syn_incolumn__divby__lenght_between_axis_values_divby_to-
tal_dend_lenght)

        dframe['%s_number' %(color)].append(np.ar-
ray(nums_along_axis))
        dframe['%s_dens' %(color)].append(denses_along_axis)

        if show_plots == True: # Verilerin grafiklendirilmesi

            plt.subplot(6,1,1)

```

```

plt.plot(check_p*px_size, nums_along_axis, c =
color, label = color)
plt.xlabel('Distance from surface (um)')
plt.ylabel('number of synapses')
plt.legend()
plt.title('Number of synapses in mice: %s'
%(folder))

if check_axis == layer_axis:
for layer in layers:
layer_place = np.where(synapses['layer']
== layer)

axis_locs = synap-
ses[check_axis].iloc[layer_place]
max_layer = max(axis_locs) * px_size

plt.axvline(x = max_layer, linestyle = '-
-', color = 'b')

plt.subplot(6,1,2)
plt.plot(check_p*px_size, denses_along_axis, c =
color, label = color)
plt.xlabel('Distance from surface (um)')
plt.ylabel('areal density')
plt.legend()
plt.title('Density of synapses in mice: %s'
%(folder))

if check_axis == layer_axis:
for layer in layers:
layer_place = np.where(synapses['layer']
== layer)

axis_locs = synap-
ses[check_axis].iloc[layer_place]
max_layer = max(axis_locs) * px_size
plt.axvline(x = max_layer, linestyle = '-
-', color = 'b')

plt.subplot(6,1,3)
plt.plot(check_p*px_size,
syn_divby_dend_lenght_along_axis,
c = color, label = color)
plt.xlabel('Distance from surface (um)')
plt.ylabel('Dendritic density (1/um)')
plt.legend()
plt.title('Density of synapses in mice: %s'
%(folder))

```

```

        if check_axis == layer_axis:
            for layer in layers:
                layer_place = np.where(synapses['layer']
== layer)

                axis_locs = synap-
ses[check_axis].iloc[layer_place]
                max_layer = max(axis_locs) * px_size
                plt.axvline(x = max_layer, linestyle = '-
-', color = 'b')

                plt.subplot(6,1,4)
                plt.plot(check_p*px_size,
syn_divby_dend_lenght_of_dominant_layer_along_axis,
                    c = color, label = color)
                plt.xlabel('Distance from surface (um)')
                plt.ylabel('Dendritic density (1/um)')
                plt.legend()
                plt.title('Density of synapses in mice: %s'
%(folder))

        if check_axis == layer_axis:
            for layer in layers:
                layer_place = np.where(synapses['layer']
== layer)

                axis_locs = synap-
ses[check_axis].iloc[layer_place]
                max_layer = max(axis_locs) * px_size
                plt.axvline(x = max_layer, linestyle = '-
-', color = 'b')

                plt.subplot(6,1,5)
                plt.plot(check_p*px_size, syn_divby_total_syn_in-
column_array,
                    c = color, label = color)
                plt.xlabel('Distance from surface (um)')
                plt.ylabel('Dendritic density (1/um)')
                plt.legend()
                plt.title('Eq. 1 in mice: %s' %(folder))

        if check_axis == layer_axis:
            for layer in layers:
                layer_place = np.where(synapses['layer']
== layer)

                axis_locs = synap-
ses[check_axis].iloc[layer_place]
                max_layer = max(axis_locs) * px_size

```



```

plt.axvline(x = max_layer, linestyle = '-
-', color = 'b')

plt.subplot(6,1,6)
plt.plot(check_p*px_size, syn_divby_total_syn_in-
column_divby_lenght_between_axis_values_divby_to-
tal_dend_lenght_array,
        c = color, label = color)
plt.xlabel('Distance from surface (um)')
plt.ylabel('Dendritic density (1/um)')
plt.legend()
plt.title('Eq. 2 in mice: %s' %(folder))

if check_axis == layer_axis:
    for layer in layers:
        layer_place = np.where(synapses['layer']
== layer)

        axis_locs = synap-
ses[check_axis].iloc[layer_place]
        max_layer = max(axis_locs) * px_size
        plt.axvline(x = max_layer, linestyle = '-
-', color = 'b')

os.chdir(main_path)
pd_dframe = pd.DataFrame(data = dframe) # Veri tabanının
kaydedilmesi

pd_dframe.to_pickle('draw_synaptic_density_volume_%s.pkl'
%(check_axis))

```

```

def get_depth_data(x_data_name, y_data_name, animal_pro-
file_name):

    '''
    Bu algoritma draw_synaptic_density_volume algoritmasındaki
    hesabın kortikal katmanlar boyunca olan hesabın seçilmesini
    sağlar ve uygun yeni bir veri seti oluşturur.
    Ayrıca kortikal katman koordinatlarını piksel ve µm cinsinden
    kontrol eder.
    Bu algoritma çalıştırılmadan önce draw_synaptic_density_volume
    algoritması x ve y yönlerinde ayrı ayrı çalıştırılmaldır.

    Girdiler:
    x_data_name: draw_synaptic_density_volume algoritmasının x
    yönünde çalıştırılmasıyla elde edilen veri seti
    y_data_name: draw_synaptic_density_volume algoritmasının y
    yönünde çalıştırılmasıyla elde edilen veri seti
    '''

```

animal\_profile\_name: İncelenen kesitin hangi fareye ait olduğu, farelerin davranış fenotiplerini, korteks kalınlıklarını gösteren veri seti. Elle oluşturulur.

Çıktı:

dframe veri seti: .pkl uzantıdır. Kortikal katmanlar boyunca piksel ve  $\mu\text{m}$  referans bilgilerini, toplam dendritik segment uzunluğunu katmandaki sinaps sayısı ve dendritik uzunluğa düşen sinaps yoğunluğunu içerir.

'''

```
import os
import numpy as np      # Kütüphanelerin yüklenmesi
import pandas as pd

colors = ['cyan', 'green', 'yellow']
invested_cols = ['number', 'dens']
x_data = pd.read_pickle(x_data_name) # x ve y yönlü veri
setlerinin yüklenmesi
y_data = pd.read_pickle(y_data_name)
mice_profile = pd.read_csv(animal_profile_name)

dframe = {}
x_cols = x_data.columns
mice_cols = mice_profile.columns

for col in mice_cols:
    dframe[col] = mice_profile[col]

dframe['check_px'] = [] # veri tabanının kurulması
dframe['cortex_depth_in_um'] = []
dframe['L1_from_um'] = []
dframe['cortex_img_depth_px'] = []
dframe['bins_um'] = []
dframe['bins_norm'] = []
dframe['total_dend_in_um'] = []

for color in colors:
    for inv_col in invested_cols:
        dframe['%s_%s' %(color, inv_col)] = []

mice = mice_profile['mouse']

L1_starts = mice_profile['L1_from_um']
cortex_depths = mice_profile['cortex_depth_in_um']
behav_profs = mice_profile['behav_prof']
```

```

for col in mice_cols:
    dframe[col] = mice_profile[col]

for mouse in mice:
    animal_where = np.where(mice == mouse)
    bin_x = x_data.iloc[animal_where].reset_index()['depth_in_um'][0]
    bin_y = y_data.iloc[animal_where].reset_index()['depth_in_um'][0]
    mouse_profile = mice_profile.iloc[animal_where].reset_index()

    L1_start = mouse_profile['L1_from_um'][0]
    cortex_depth = mouse_profile['cortex_depth_in_um'][0]

    behav_prof = mouse_profile['behav_prof'][0]

    if max(bin_x) > max(bin_y): # Uygun analiz yönünün
seçilmesi
        use_data_for_depth = x_data.iloc[animal_where].reset_index()
        len_bins = len(bin_x)
    else:
        use_data_for_depth = y_data.iloc[animal_where].reset_index()
        len_bins = len(bin_y)

    L = np.linspace(0, 1, len_bins)

    if L1_start != 0:
        bins = use_data_for_depth['depth_in_um'][0]
        norm_bins = bins / (cortex_depth + L1_start)
    else:
        bins = use_data_for_depth['depth_in_um'][0]
        norm_bins = bins / cortex_depth

    dframe['bins_um'].append(bins)
    dframe['bins_norm'].append(norm_bins)
    dframe['total_dend_in_um'].append(use_data_for_depth['total_dend_in_um'][0])
    dframe['check_px'].append(use_data_for_depth['check_px'][0])

    for color in colors: # Doğru sinaps sayılarının ve yoğunluk verilerinin aktarılması
        for inv_col in invested_cols:

```

```

        color_val = use_data_for_depth['%s_%s' %(color,
inv_col)][0]
        color_val_interp = np.interp(L, norm_bins,
color_val)
        dframe['%s_%s' %(color, inv_col)].ap-
pend(color_val)

```

```

# Veri setinin kaydedilmesi
pd_dframe = pd.DataFrame(data = dframe)
pd_dframe.to_pickle('depth_data.pkl')

```

```

def get_segment_center_of_mass(filepath_from_main,
                               mice_data_file_name,
                               px_sizem = 0.103, show_plots =
False):
    '''
    Bu algoritma dendritik segmentlerin ağırlık merkezlerini
    bulur ve onları bir veri setine kaydeder.
    Tüm nodlar özdeş olduğu için ağırlık merkezi, basitçe nod
    koordinatlarının aritmetik ortalaması olarak hesaplanmıştır.
    Bu bilgi dendritik hangi kortikal katmana ait olduğunun
    tespitinde kullanılacaktır.
    Bu algoritma çalıştırılmadan önce get_branch_lengths algorit-
    ması aktiflenmelidir.

    Girdiler:
    filepath_from_main: Ana dosyadan itibaren hedeflenen dosyaya
    uzanan hedef yolu.
    mice_data_file_name: .csv uzantılıdır. Elle hazırlanmış bu
    dosya kesitlerin hangi fareye ait olduğunu, farelerin davranış fe-
    notiplerini, kesit kalınlıklarını ve eğer içermiyorsa 1. katman-
    dan ne kadar kaybettiğinin bilgisini içermelidir.
    px_sizem: Görüntü alımı sırasında ayarlanan µm cinsinden xy
    piksel mesafesi

    Çıktı:
    dframe: Kesiti incelenen fare bilgisini, davranış fenotipini,
    segmentin etiketi, uzunluğu ve ağırlık merkezinin 3 boyutlu
    koordinatlarını gösteren veri seti.

    '''

import os
import numpy as np      # Kütüphanelerin yüklenmesi
import pandas as pd

```

```

import matplotlib.pyplot as plt

mice_data = pd.read_csv('%s.csv' %(mice_data_file_name),
header = 0)

dframe = {'mouse': [], # Veri setinin kurulması
          'behav_prof': [],
          'segment': [],
          'segment_length': [],
          'com_x': [],
          'com_y': [],
          'com_z': []}

main_path = os.getcwd() # İşlem dosya merkezinin
değiştirilmesi
folder_path = os.path.join(main_path, filepath_from_main)
os.chdir(folder_path)

folder_list = os.listdir()

for folder in folder_list:
    print(folder)

    lengths = []

    coms_x = []
    coms_y = []
    coms_z = []

    if '.' in folder:
        continue

    file_path = os.path.join(folder_path, folder)
    os.chdir(file_path)

    file_list = os.listdir()

    for file in file_list:

        if '.swc' in file:
            if 'filtered.swc' in file:
                reconst_file_name = file
            else:
                continue
        else:
            continue

```

```

reconst = pd.read_csv(reconst_file_name, header = 0, sep
= ' ')

segments = np.unique(reconst['segment'])
dframe['segment'].append(segments)
dframe['mouse'].append(folder)

mouse_where = np.where(mice_data['mouse'] == folder)
behav = mice_data.iloc[mouse_where].reset_index()['be-
hav_prof'][0]
dframe['behav_prof'].append(behav)

for seg in segments: # Her bir segment için ağırlık
merkezinin hesaplanması

    segment_where = np.where(reconst['segment'] == seg)
    segment = reconst.iloc[segment_where]

    length, _ = get_branch_lengths(segment, px_size =
px_sizem, show_plot = show_plots)
    lengths = np.append(lengths, length)

    com_x = np.mean(segment['x'])
    com_y = np.mean(segment['y'])
    com_z = np.mean(segment['z'])

    coms_x = np.append(coms_x, com_x)
    coms_y = np.append(coms_y, com_y)
    coms_z = np.append(coms_z, com_z)

    dframe['com_x'].append(coms_x)
    dframe['com_y'].append(coms_y)
    dframe['com_z'].append(coms_z)
    dframe['segment_length'].append(lengths)

os.chdir(main_path)

dframe = pd.DataFrame(data = dframe) # Veri setinin
kaydedilmesi
dframe.sort_values('mouse')

dframe.to_pickle('segment_center_of_mass.pkl')

```

```

# connection_data, depth_data ve center_of_mass verisetleri
kullanılarak her bir katmandaki sinapsların koordinatları ve
hacimleri çıkarıldı.
# Sonrasında elde edilen veri synapse_data.pkl olarak kaydedildi.

import os
import numpy as np      # Kütüphanelerin yüklenmesi
import pandas as pd

data = pd.read_pickle('connection_data+depth_data+segment_center_of_mass.pkl') # Sinaptik ve dendritik verilerin yüklenmesi
add_to_col = data['cortex_img_depth_px'] * (data['L1_from_um'] / data['cortex_depth_in_um']) # 1. katmanı olmayan kesitler için düzeltme mesafesinin hesaplanması
filepath_from_main = 'main'
px_size = 0.103

main_path = os.getcwd()
folder_path = os.path.join(main_path, filepath_from_main)
os.chdir(folder_path)

folder_list = os.listdir()

column_syn_colors = []
column_syn_x_coords = []
column_syn_y_coords = []
column_syn_z_coords = []
column_syn_volumes = []

for folder in folder_list: # Her bir kesit için sinaps, dendrit verisetlerinin yüklenmesi
    if '.' in folder:
        continue

    print(folder)

    mouse_where = np.where(data['mouse'] == folder)

    file_path = os.path.join(folder_path, folder)
    os.chdir(file_path)

    dendrites, synapses_isq, synapses_marker =
get_swc_marker_isq_data()

    x_max = max(dendrites['x'])
    y_max = max(dendrites['y'])

```

```

if x_max > y_max:
    look = 'x'
else:
    look = 'y'
adding = add_to_col.iloc[mouse_where]
adding = adding[adding.index[0]]
synapses_isq[look] = synapses_isq[look] + adding
check_px = data['check_px'].iloc[mouse_where]
check_px = check_px[check_px.index[0]]

syn_colors = []
syn_x_coords = []
syn_y_coords = []
syn_z_coords = []
syn_volume = []

for i in np.arange(len(check_px) - 1): # her bir kesit için
    sinaps koordinat ve hacim tespiti
    syn_where = np.where((synapses_isq[look] > check_px[i]) &
(synapses_isq[look] < check_px[i+1]))
    synapses_in_layer = synapses_isq.iloc[syn_where]
    if len(synapses_in_layer) > 0:
        syn_color = np.hstack(synapses_in_layer['color'])
        syn_x = np.hstack(synapses_in_layer['x'])
        syn_y = np.hstack(synapses_in_layer['y'])
        syn_z = np.hstack(synapses_in_layer['z'])
        syn_vol = np.hstack(synapses_in_layer['volume']) *
(px_size)**3
    else:
        syn_color = np.array([])
        syn_x = np.array([])
        syn_y = np.array([])
        syn_z = np.array([])
        syn_vol = np.array([])

    syn_colors.append(syn_color)
    syn_x_coords.append(syn_x)
    syn_y_coords.append(syn_y)
    syn_z_coords.append(syn_z)
    syn_volume.append(syn_vol)

column_syn_colors.append(syn_colors)
column_syn_x_coords.append(syn_x_coords)
column_syn_y_coords.append(syn_y_coords)
column_syn_z_coords.append(syn_z_coords)
column_syn_volumes.append(syn_volume)

```



```
dframe = {'mouse': data['mouse'], # verisetinin oluşturulması
          'behav_prof': data['behav_prof'],
          'color': column_syn_colors,
          'syn_x': column_syn_x_coords,
          'syn_y': column_syn_y_coords,
          'syn_z': column_syn_z_coords,
          'vol': column_syn_volumes}
```

```
dframe = pd.DataFrame(data = dframe)
```

```
dframe.to_pickle('synapse_data.pkl')
```

```
def get_segment_numbers_in_layers(synapse_data, segment_colname,
                                  syn_num_colname, layer_name,
                                  get_colorful = True):

    """
    Bu algoritma her bir kortikal katmandaki farklı renkli kombinasyonlara ait segmentlerin (örn. VH ve BLA içerenler) uzunluklarını ve segment etiket numaralarını çeker.

    Girdiler:
    Önceki işlemlerde oluşturulmuş verisetleri: connection_data.pkl, depth_data.pkl, segment_center_of_mass.pkl, synapse_data.pkl
    segment_colname: segment tipi, değişkenler: cyan_only_segments; green_only_segments; yellow_only_segments; cyan_green_not_yellow_segments; cyan_yellow_not_green_segments; green_yellow_not_cyan_segments; cyan_green_yellow_segments
    syn_num_colname: veri setinde ilgili segmentin sahip olduğu sinaps sayısını veren değişken ismi
    layer_name: incelenen kortikal katman
    get_colorful: bu incelemenin renk renk yapılmayacağını belirler. Eğer 'False' olursa bölgedeki tüm segmentlerin uzunluk verisetini oluşturur.

    Çıktılar:
    dframe: İlgili segment etiketlerini içeren veri seti
    dframe2: İlgili segmentlerde bulunan sinaps sayısını veren veri seti
    """

    segments_in_layer_all = []
    segment_lengths_in_layer_all = []
    segment_lengths_in_layer_means_all = []
```

```

add_to_col = synapse_data['cortex_img_depth_px'] * (syn-
apse_data['L1_from_um'] / synapse_data['cortex_depth_in_um'])
for i in np.arange(len(synapse_data)):
    com_x_max = max(synapse_data['com_x'][i])
    com_y_max = max(synapse_data['com_y'][i])
    check_px = synapse_data['check_px'][i]

    if com_x_max > com_y_max:
        use = 'com_x'
    else:
        use = 'com_y'

    com_coords = synapse_data[use][i] + add_to_col[i]
    segments_in_layer_one = []
    segment_lengths_in_layer_one = []
    segment_lengths_in_layer_means = []

    for j in np.arange(len(check_px) - 1):
        com_where = np.where((com_coords < check_px[j+1]) &
(com_coords > check_px[j]))
        segments_in_layer = synapse_data['seg-
ment'][i][com_where]
        segments_in_layer_one.append(segments_in_layer)

        segment_lengths_in_layer = synapse_data['seg-
ment_length'][i][com_where]

        segment_lengths_in_layer_mean = np.mean(seg-
ment_lengths_in_layer)
        segment_lengths_in_layer_means = np.append(seg-
ment_lengths_in_layer_means,
                                                    seg-
ment_lengths_in_layer_mean)

        segment_lengths_in_layer_one.append(seg-
ment_lengths_in_layer)

        segments_in_layer_all.append(segments_in_layer_one)
        segment_lengths_in_layer_all.append(seg-
ment_lengths_in_layer_one)
        segment_lengths_in_layer_means_all.append(seg-
ment_lengths_in_layer_means)

    if get_colorful == False:
        dframe = {'segment_lengths_in_layer_all': seg-
ment_lengths_in_layer_all}
        dframe = pd.DataFrame(data = dframe)

```

```

    return dframe

    colorful_segments_in_layer_all = []
    syn_num_colorful_segments_in_layer_all = []

    for i in np.arange(len(synapse_data)):
        colorful_segments = data[segment_colname][i]
        if layer_name == 'L1':
            segments_in_layer = segments_in_layer_all[i][0]
        elif layer_name == 'L2/3':
            segments_in_layer = np.hstack(segments_in_layer_all[i][1:3])
        elif layer_name == 'L5':
            segments_in_layer = np.hstack(segments_in_layer_all[i][3:8])
        elif layer_name == 'L6':
            segments_in_layer = np.hstack(segments_in_layer_all[i][8:])

        control = np.isin(colorful_segments, segments_in_layer)

        colorful_segments_in_layer = colorful_segments[control]

        if len(data[syn_num_colname][i]) == 0:
            syn_num_colorful_segments_in_layer = np.array([])
        else:
            syn_num_colorful_segments_in_layer = data[syn_num_colname][i][control]

        colorful_segments_in_layer_all.append(colorful_segments_in_layer)
        syn_num_colorful_segments_in_layer_all.append(syn_num_colorful_segments_in_layer)

    dframe = {'%s_in_layer_all' % (segment_colname): colorful_segments_in_layer_all}
    dframe2 = {'syn_num_%s_in_layer_all' % (segment_colname): syn_num_colorful_segments_in_layer_all}

    dframe = pd.DataFrame(data = dframe)
    dframe2 = pd.DataFrame(data = dframe2)

    return dframe, dframe2

```

```

# get_segment_numbers_in_layers tüm katman ve segment türleri
için ayrı ayrı çalıştırılmalıdır.
# Sonrasında birleştirilip tek bir veri setine dönüştürülebilir.
# Aşağıda örnek olması amacıyla 1. katman için yapılan işlem
gösterilmiştir.
# Sonrasında 1. katmandaki veri setleri birleştirilerek
kaydedilmiştir.

data = pd.read_pickle('connection_data+depth_data+segment_center_of_mass+synapse_data.pkl')
data = data.sort_values('behav_prof', ignore_index = True)

cyan_only_segments_L1, cyan_only_syn_num_L1 = get_segment_numbers_in_layers(data,

    'cyan_only_segments',

    'syn_num_cyan_only_segments', 'L1')

green_only_segments_L1, green_only_syn_num_L1 = get_segment_numbers_in_layers(data,

    'green_only_segments',

    'syn_num_green_only_segments', 'L1')

yellow_only_segments_L1, yellow_only_syn_num_L1 = get_segment_numbers_in_layers(data,

    'yellow_only_segments',

    'syn_num_yellow_only_segments', 'L1')

cyan_green_segments_L1, cyan_green_syn_num_L1 = get_segment_numbers_in_layers(data,

    'cyan_green_not_yellow_segments',

    'syn_num_cyan_green_not_yellow_segments',

    'L1')

cyan_yellow_segments_L1, cyan_yellow_syn_num_L1 = get_segment_numbers_in_layers(data,

    'cyan_yellow_not_green_segments',

```

```

        'syn_num_cyan_yellow_not_green_segments',
        'L1')

green_yellow_segments_L1, green_yellow_syn_num_L1 = get_seg-
ment_numbers_in_layers(data,

        'green_yellow_not_cyan_segments',

        'syn_num_green_yellow_not_cyan_segments',

        'L1')

cyan_green_yellow_segments_L1, cyan_green_yellow_syn_num_L1 =
get_segment_numbers_in_layers(data,

        'cyan_green_yellow_segments',

        'syn_num_cyan_green_yellow_segments',

        'L1')

dframe_L1 = pd.concat([data['mouse'],
                        data['behav_prof'],
                        cyan_only_segments_L1,
                        green_only_segments_L1,
                        yellow_only_segments_L1,
                        cyan_green_segments_L1,
                        cyan_yellow_segments_L1,
                        green_yellow_segments_L1,
                        cyan_green_yellow_segments_L1], axis = 1)

dframe_L1.to_pickle('L1_color_pos_segments.pkl')

```

```

def get_successive_synaptic_distances(synapses, dendrite, px_size
= 0.103,
                                     cyan = True, green = True,
yellow = True,
                                     mode = 'dendritic'):

    '''
    Bu algoritma bir dendritik segment üstündeki iki komşu sinaps
    arasındaki mesafeyi öklid veya dendrit üstü bulur.
    Mesafeyi µm cinsinden kaydeder. Bu algoritma hem peşpeşe gelen
    farklı renkteki iki sinaps mesafesini hem de aynı renkte komşu
    iki sinaps arasındaki mesafeyi tespit edebilir.
    '''

```

Bu süreçte sinapsların nasıl dizildiği de kaydedilir.

Girdiler:

```
synapses: bir segment üzerindeki sinapslar. Önceki veri
setlerinden elde edilebilir.
dendrite: dendrit rekonstrüksiyon ağacındaki bir segment
px_size: Görünütüleme esnasında ayarlanan xy piksel mesafesi
cyan: VTA sinapslarının incelenip incelenmeyeceği, veri tipi:
binary
green: VH sinapslarının incelenip incelenmeyeceği, veri tipi:
binary
yellow: BLA sinapslarının incelenip incelenmeyeceği, veri tipi:
binary
mode: hesaplamanın öklid veya dendrit üstü olacağını irdeler.
'euclidean' veya 'dendritic olabilir'
```

Örnek kullanım: cyan = True, green = False ve yellow = True olursa, bir dendritik segment üzerindeki VTA ve BLA sinapsları akış halinde incelenir.

Burada hem VTA-VTA, BLA-BLA ve VTA-BLA sinaps mesafeleri hesaplanır. Ancak VH hesap dışı tutulur.

Çıktılar:

```
syn_distances: µm cinsinden sinaps mesafeleri
```

```
'''
```

```
import os # Kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd

if np.unique(synapses['segment']) != np.unique(dendrite['segment']):
    raise 'segments of the structures needs to be same'

syn_distances = []

logic = np.array([cyan, green, yellow]) # İncelenecek rengin
renk ismini yazmadan sadece mantık operasyonları belirtmek suretiyle tespiti
cyan_name = f'{cyan=}'.split('=')[0]
green_name = f'{green=}'.split('=')[0]
yellow_name = f'{yellow=}'.split('=')[0]

names = np.array([cyan_name, green_name, yellow_name])

colors = names[logic]
```

```

synapses_sorted = synapses.sort_values('real_nearest_node')
color_where = np.isin(synapses['color'], colors)
synapses_sorted = synapses_sorted[color_where]

indexes = synapses_sorted.index

if mode == 'dendritic': # Dendrit üstü intersinaptik mesafe
hesabi
    for i in np.arange(len(synapses_sorted) - 1):

        syn = indexes[i]
        syn_next = indexes[i+1]

        node_id = synapses_sorted['real_nearest_node'][syn]
        next_node_id = synapses_sorted['real_nearest_node']
est_node'][syn_next]

        node_where_dend = np.where(dendrite['#id'] ==
node_id)
        node = dendrite.iloc[node_where_dend]
        next_node_where_dend = np.where(dendrite['#id'] ==
next_node_id)
        next_node = dendrite.iloc[next_node_where_dend]

        '''
added on 23.12.23
        '''

        if len(next_node) == 0 or len(node) == 0:
            syn_distances = []
            return syn_distances

        #print(node['x'][node.index[0]])

        dx = next_node['x'][next_node.index[0]] -
node['x'][node.index[0]]
        dy = next_node['y'][next_node.index[0]] -
node['y'][node.index[0]]
        dz = next_node['z'][next_node.index[0]] -
node['z'][node.index[0]]

        #print('dx:', dx, 'dy', dy, 'dz', dz)

        dsyn = np.sqrt(dx**2 + dy**2 + dz**2)

        syn_distances = np.append(syn_distances, dsyn)

```

```

elif mode == 'euclidean': # Öklid intersinaptik mesafe hesabı
    for i in np.arange(len(synapses_sorted) - 1):

        syn = indexes[i]
        syn_next = indexes[i+1]

        dx = synapses_sorted['x'][syn_next] - synapses_sorted['x'][syn]
        dy = synapses_sorted['y'][syn_next] - synapses_sorted['y'][syn]
        dz = synapses_sorted['z'][syn_next] - synapses_sorted['z'][syn]

        dsyn = np.sqrt(dx**2 + dy**2 + dz**2)

        syn_distances = np.append(syn_distances, dsyn)

if len(syn_distances) == 0:
    return syn_distances

syn_distances = syn_distances * 0.103

return syn_distances

```

```

def get_syn_dend_vector(synapses, dendrite, px_size = 0.103,
                       cyan = True, green = True, yellow = True,
                       mode = 'normalized'):

    """
    Bu algoritma bir dendritik segmentin noda en yakın sinaps
    sayılarını hesaplar.
    Girdiler:
    synapses: bir segment üzerindeki sinapslar. Önceki veri
    setlerinden elde edilebilir.
    dendrite: dendrit rekonstrüksiyon ağacındaki bir segment
    px_size: Görünütüleme esnasında ayarlanan xy piksel mesafesi
    cyan: VTA sinapslarının incelenip incelenmeyeceği, veri tipi:
    binary
    green: VH sinapslarının incelenip incelenmeyeceği, veri tipi:
    binary
    yellow: BLA sinapslarının incelenip incelenmeyeceği, veri
    tipi: binary
    mode: hesaplamının segment uzunluğuna normalize edilip
    edilmeyeceğini irdeler. 'normalized' yazılmazsa normalizasyon
    gerçekleşmez.
    """

```



```

Çıktılar:
syn_counts: Segment üzerindeki her bir nod üzerindeki sinaps
sayısı
dend_distance_array: 0'dan başlayarak dendritik segmentin son
uzunluğuna uzanan dendritik uzunluk zinciri, dendritik segment
üzerinde kümelenmenin olup olmadığının incelenmesi için
kullanılabilir.

'''

import os # Kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd

if np.unique(synapses['segment']) != np.unique(dendrite['seg-
ment']):
    raise 'segments of the structures needs to be same'

dend_distance_array = 0
dist = 0
syn_counts = []

logic = np.array([cyan, green, yellow]) # İncelenecek rengin
renk ismini yazmadan sadece mantık operasyonları belirtmek sure-
tiyle tespiti
cyan_name = f'{cyan=}'.split('=')[0]
green_name = f'{green=}'.split('=')[0]
yellow_name = f'{yellow=}'.split('=')[0]

names = np.array([cyan_name, green_name, yellow_name])

colors = names[logic]

synapses_sorted = synapses.sort_values('real_nearest_node')
color_where = np.isin(synapses['color'], colors)
synapses_sorted = synapses_sorted[color_where]

for i in dendrite['#id']:
    num_syn_on_node = len(np.where(synap-
ses_sorted['real_nearest_node'] == i)[0])
    syn_counts = np.append(syn_counts, num_syn_on_node) # her
bir noda en yakın sinaps sayısının hesaplanması

indexes = dendrite.index

for i in np.arange(len(dendrite)-1): # İki sinaps arası
mesafe zincirinin hesaplanması

```

```

node = indexes[i]
next_node = indexes[i+1]

dx = dendrite['x'][next_node] - dendrite['x'][node]
dy = dendrite['y'][next_node] - dendrite['y'][node]
dz = dendrite['z'][next_node] - dendrite['z'][node]

dL = np.sqrt(dx**2 + dy**2 + dz**2)
dist += dL
dend_distance_array = np.append(dend_distance_array,
dist)

dend_distance_array = dend_distance_array * 0.103

if mode == 'normalized':
    dend_distance_array = dend_distance_array/dend_dis-
tance_array[-1]

return (syn_counts, dend_distance_array)

```

```

def get_density_series_dframe(segment_dframe, column_name,
                             layer_info, content_info,
color_info,
                             filepath_from_main, px_size):

    '''
    Bu algoritma get_syn_dend_vector ve get_successive_synap-
tic_distances algoritmalarının sonuçlarını
    bir katman ve bir segment türü için birleştirip bir veri setine
    dönüştürür.

    Algoritmanın çalışması için get_branch_lengths ve

    Girdiler:
    segment_dframe: Kortikal katmandaki dendritik segment, sinaptik
bağlantı verilerini içeren veri seti
    column_name: İncelenen segment tipinin özel veri etiket ismi,
veri tabanının incelenmesi tavsiye edilir.
    layer_info: katman bilgisi
    content_info: İlgili segmentte kaç çeşit sinaps bulunduğu bilg-
isi. 'single', 'double' veya 'triple' olabilir.
    color_info: İlgili segmentte bulunan sinaps renkleri.
    filepath_from_main: Ana dosyadan itibaren hedeflenen dosyaya
uzanan hedef yolu.
    px_size: Görünütüleme esnasında ayarlanan xy piksel mesafesi

```

Çıktılar:

dframe: İlgili segmentin türü, üzerindeki sinapsların renkleri, koordinatları, birbirleri arasındaki mesafeleri, her bir çeşit sinapstan kaç tane olduğu bilgisini içeren veri seti

```
'''
import os # Kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd

main_path = os.getcwd()
folder_path = os.path.join(main_path, filepath_from_main)
os.chdir(folder_path)
folder_list = os.listdir()

mouse_series = []
behav_series = []
layer_series = []

syn_num = []
cyan_syn_num = []
green_syn_num = []
yellow_syn_num = []

content_series = []
color_series = []
color_sequences = []
syn_x = []
syn_y = []
syn_z = []
syn_vol = []

seg_lengths = []
dend_x = []
dend_y = []
dend_z = []
syn_count_on_node = []
intersyn_dist = []
segment_nums = []

for folder in folder_list:
    if '.' in folder:
        continue

    print(folder)
```

```

mouse_where = np.where(segment_dframe['mouse'] == folder)
behav = segment_dframe['behav_prof'].iloc[mouse_where]
idx = segment_dframe.iloc[mouse_where].index
idx = idx[0]

behav = segment_dframe['be-
hav_prof'].iloc[mouse_where][idx]

file_path = os.path.join(folder_path, folder)
os.chdir(file_path)

dendrites, synapses_isq, synapses_marker =
get_swc_marker_isq_data() # Dendritik rekonstrüksiyon ve sinaps
verisetlerinin yüklenmesi

if len(segment_dframe[col-
umn_name].iloc[mouse_where][idx]) == 0:
    continue

for i in segment_dframe[col-
umn_name].iloc[mouse_where][idx]:

    seg_where = np.where(dendrites['segment'] == i)
    syn_where = np.where(synapses_isq['segment'] == i)

    segment = dendrites.iloc[seg_where]
    synapses = synapses_isq.iloc[syn_where]
    synapses = synapses.sort_values('real_nearest_node')

    seg_length, _ = get_branch_lengths(segment, px_size =
px_size, show_plot=False)
    seg_length = seg_length[0]

    syn_counts, dend_distance_array = get_syn_dend_vec-
tor(synapses, segment, # en yakın nod üzerindeki sinaps
sayılarının intersinaptik mesafelerin çıkarılması

px_size = px_size,

cyan = True, green = True,

yellow = True, mode = '')

    syn_distances = get_successive_synaptic_dis-
tances(synapses, segment, px_size = px_size, # Artan inter-
sinaptik mesafe zincirinin oluşturulması

```

```

n = True, green = True, yellow = True,
e = 'dendritic')

        mouse_series.append(folder) # Veri setindeki element-
lerin oluşturulması
        behav_series.append(behav)
        layer_series.append(layer_info)

        syn_num.append(len(synapses))

        colors_seg, color_count = np.unique(synapses['col-
or'], return_counts = True)

        if len(color_count[colors_seg == 'cyan']) == 0:
            cyan_syn_num.append(0)
        else:
            cyan_syn_num.append(color_count[colors_seg ==
'cyan'][0])

        if len(color_count[colors_seg == 'green']) == 0:
            green_syn_num.append(0)
        else:
            green_syn_num.append(color_count[colors_seg ==
'green'][0])

        if len(color_count[colors_seg == 'yellow']) == 0:
            yellow_syn_num.append(0)
        else:
            yellow_syn_num.append(color_count[colors_seg ==
'yellow'][0])

        content_series.append(content_info)
        color_series.append(color_info)
        color_sequences.append(list(synapses['color']))

        syn_vol.append(np.array(synapses['volume']))

        syn_x.append(np.array(synapses['x']))
        syn_y.append(np.array(synapses['y']))
        syn_z.append(np.array(synapses['z']))

```

```

    seg_lengths.append(seg_length)
    dend_x.append(np.array(segment['x']))
    dend_y.append(np.array(segment['y']))
    dend_z.append(np.array(segment['z']))

    syn_count_on_node.append(syn_counts)
    intersyn_dist.append(syn_distances)
    segment_nums.append(i)

dframe = {'mouse': mouse_series, # Kaydedilebilir veri setine
dönüştürme
        'behav_prof': behav_series,
        'layer': layer_series,
        'segment': segment_nums,
        'syn_num': syn_num,
        'cyan_syn_num': cyan_syn_num,
        'green_syn_num': green_syn_num,
        'yellow_syn_num': yellow_syn_num,

        'content': content_series,
        'color': color_series,
        'color_sequence': color_sequences,
        'syn_vol': syn_vol,
        'syn_x': syn_x,
        'syn_y': syn_y,
        'syn_z': syn_z,
        'seg_length': seg_lengths,
        'dend_x': dend_x,
        'dend_y': dend_y,
        'dend_z': dend_z,
        'syn_count_on_node': syn_count_on_node,
        'intersyn_dist': intersyn_dist}

dframe = pd.DataFrame(data = dframe)
return dframe

# get_density_series_dframe algoritması her bir katman ve her bir
segment tipi için ayrı ayrı uygulanır.
# Sonrasında veri setleri üst üste getirilerek kaydedilir.
# Buradan çıkan veri çok renkli dendritik segmentlerin incelen-
mesi için kullanılmıştır.
# Aşağıda 1. katman için yapılmış örnek bir uygulama
gösterilmiştir.

data = pd.read_pickle('connection_data+depth_data+segment_cen-
ter_of_mass+synapse_data.pkl')
data = data.sort_values('behav_prof', ignore_index = True)

```

```

data_L1_segments = pd.read_pickle('L1_color_pos_segments.pkl')
data_L23_segments = pd.read_pickle('L23_color_pos_segments.pkl')
data_L5_segments = pd.read_pickle('L5_color_pos_segments.pkl')
data_L6_segments = pd.read_pickle('L6_color_pos_segments.pkl')

cyan_only_L1 = get_density_series_dframe(data_L1_segments,
'cyan_only_segments_in_layer_all',
'L1', 'single', 'cyan',
'main', px_size = 0.103)

green_only_L1 = get_density_series_dframe(data_L1_segments,
'green_only_segments_in_layer_all',
'L1', 'single', 'green',
'main', px_size = 0.103)

yellow_only_L1 = get_density_series_dframe(data_L1_segments,
'yellow_only_segments_in_layer_all',
'L1', 'single', 'yel-
low',
'main', px_size = 0.103)

cyan_green_L1 = get_density_series_dframe(data_L1_segments,
'cyan_green_not_yellow_segments_in_layer_all',
'L1', 'double', 'cyan-
green',
'main', px_size =
0.103)

cyan_yellow_L1 = get_density_series_dframe(data_L1_segments,
'cyan_yellow_not_green_segments_in_layer_all',
'L1', 'double', 'cyan-
yellow',
'main', px_size =
0.103)

green_yellow_L1 = get_density_series_dframe(data_L1_segments,
'green_yellow_not_cyan_segments_in_layer_all',
'L1', 'double', 'green-
yellow',
'main', px_size =
0.103)

cyan_green_yellow_L1 = get_density_series_dframe(data_L1_seg-
ments, 'cyan_green_yellow_segments_in_layer_all',
'L1', 'triple', 'cyan-
green-yellow',

```

```

                                                                 'main', px_size =
0.103)

L1_syn_density_allcolors = pd.concat([cyan_only_L1,
                                       green_only_L1,
                                       yellow_only_L1,
                                       cyan_green_L1,
                                       cyan_yellow_L1,
                                       green_yellow_L1,
                                       cyan_green_yellow_L1], ignore_index = True)

column_syn_density_allcolors = pd.concat([L1_syn_density_allcol-
ors,
                                           L23_syn_density_allcol-
ors,
                                           L5_syn_density_allcol-
ors,
                                           L6_syn_density_allcol-
ors], ignore_index = True)

column_syn_density_allcolors.to_pickle('synapses_seg-
ments_all_layers.pkl')

```

```

# 5b katmanı için BLA sinaps oranlarının gruplar arası
karşılaştırması ve sosyal etkileşim oranıyla orantılanması Şekil
4.8. ve Şekil 4.9.
# Bu algoritma için her bir grafiğin ayrı düzenlemeye ihtiyaç
duyması sebebiyle herhangi bir fonksiyon oluşturulmadı.
# Algoritma her bir subkortikal beyin bölgesinden kaynaklı sin-
apsların oranlarının katmanlar boyunca bulunması ve
# onların stres duyarlılık fenotipleri arasında karşılaştırılması
ve sosyal etkileşim oranıyla ilişkilendirilmesini gösterir.
# Diğer katmanlar da uygun değişkenler oluşturularak
incelenmiştir.

```

```

import os # Kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats
from scipy.stats import mannwhitneyu as mwu
from scipy.stats import kruskal, spearmanr, linregress
from scipy.stats import f_oneway as anova

import statsmodels.api as sm

```



```

data = pd.read_pickle('connection_data+depth_data+segment_center_of_mass+synapse_data.pkl') # Veri setinin yüklenmesi
data = data.sort_values('behav_prof', ignore_index = True)

sus_where = np.where(data['behav_prof'] == 'sus') # Veri de hızlı bulma için kullanılan yardımcı değişkenler
res_where = np.where(data['behav_prof'] == 'res')
nai_where = np.where(data['behav_prof'] == 'nai')
stress_where = np.where((data['behav_prof'] == 'res') | (data['behav_prof'] == 'sus'))

stress_prof = np.array(['nai']* len(data), dtype = 'object')
stress_prof[stress_where] = 'stress'

black_to_white = sns.color_palette("gray", n_colors=12)

behav_palette = sns.color_palette([black_to_white[-1], black_to_white[8], black_to_white[0]])
stress_palette = sns.color_palette([black_to_white[-1], black_to_white[3]])

sum_syn_layers = np.nansum([np.vstack(data['yellow_number']), # Katmandaki sinaps sayılarının matrisine dönüştürülmesi
                             np.vstack(data['cyan_number']),
                             np.vstack(data['green_number'])],
axis = 0)

cyan_numbers = np.vstack(data['cyan_number'])
green_numbers = np.vstack(data['green_number'])
yellow_numbers = np.vstack(data['yellow_number'])

f = np.array(['nai'] * 15, dtype = 'object') # Şekil oluşturmada kullanılacak etiket ve işaret düzenlemeleri
f[data['behav_prof'] == 'nai'] = 'Naive'
f[data['behav_prof'] == 'res'] = 'Resilient'
f[data['behav_prof'] == 'sus'] = 'Susceptible'

g = np.array(['nai'] * 15, dtype = 'object')
g[data['behav_prof'] == 'nai'] = 'Nai.'
g[data['behav_prof'] == 'res'] = 'Res.'
g[data['behav_prof'] == 'sus'] = 'Sus.'

markdict={'Nai.': 'o', 'Res.': '^', 'Sus.': 'v'}

h = np.array(['nai'] * 15, dtype = 'object')
h[data['behav_prof'] == 'nai'] = 'Nai.'

```

```

h[data['behav_prof'] == 'res'] = 'Str.'
h[data['behav_prof'] == 'sus'] = 'Str.'

yellow_perc_layer_L5v = (yellow_numbers[:,4] / sum_syn_layers[:,4]) * 100 # 5b katmanındaki BLA sinaps yüzdesinin tüm kesitler için hesaplanması
means = yellow_perc_layer_L5v

sus = means[sus_where]
res = means[res_where]
nai = means[nai_where]
stress = means[stress_where]

print(kruskal(sus,res,nai, nan_policy = 'omit')) # Kruskal-Wallis ve MWU testlerinin uygulanması
print('sus vs res, ', mwu(sus,res, nan_policy = 'omit'))
print('sus vs nai, ', mwu(sus,nai, nan_policy = 'omit'))
print('res vs nai, ', mwu(res,nai, nan_policy = 'omit'))
print('stress vs nai, ', mwu(stress,nai, nan_policy = 'omit'))

print(' ')

_, p_sus_res = mwu(sus,res, nan_policy = 'omit')
_, p_sus_nai = mwu(sus,nai, nan_policy = 'omit')
_, p_nai_res = mwu(nai,res, nan_policy = 'omit')

plt.figure(figsize = (4,6), dpi = 600) # Stres naif, stres dirençli ve stres duyarlı fareler için boks torbası grafiklerinin oluşturulması; Şekil 4.8.

boxplot = sns.boxplot(x = g[nai_where], y = means[nai_where],
                      capwidths = 0.3,
                      color = behav_palette[0],
                      linewidth = 1.5, linecolor = 'k',
                      fliersize = 0, width = 0.45, gap = 2)

boxplot = sns.boxplot(x = g[res_where], y = means[res_where],
                      capwidths = 0.3,
                      color = behav_palette[1],
                      linewidth = 1.5, linecolor = 'k',
                      fliersize = 0, width = 0.45, gap = 2)

boxplot = sns.boxplot(x = g[sus_where], y = means[sus_where],
                      capwidths = 0.3,
                      color = behav_palette[2],
                      linewidth = 1.5, linecolor = 'k',
                      fliersize = 0, width = 0.45, gap = 2,

```

```

        medianprops = dict(color = 'w'))

for patch in boxplot.patches:
    fc = patch.get_facecolor()
    patch.set_facecolor(mpl.colors.to_rgba(fc, 0.85))

stripplot = sns.swarmplot(x = g, y = means, size = 8.5,
                          edgecolor = 'k',
                          color = 'w', linewidth = 1.5)

for axis in ['bottom', 'left']:
    boxplot.spines[axis].set_linewidth(1.5)

boxplot.tick_params(width= 1.5)

boxplot.set(xlabel = None)

plt.xticks(fontsize = 25)

plt.ylabel('Percentage (%)', fontsize = 30)
plt.ylim(bottom = -1, top = 102)
plt.yticks(fontsize = 25)

sns.despine()

plt.figure(figsize = (2,6), dpi = 300) # Stres naif, kronik
strese maruz kalan fareler için boks torbası grafiklerinin
oluşturulması
boxplot = sns.boxplot(x = h, y = means,
                     palette = stress_palette,
                     linewidth = 1, fliersize = 0)
stripplot = sns.swarmplot(x = h, y = means, size = 6.5, edgecolor
= 'k',
                          color = 'w', linewidth = 1.5)

boxplot.set(xlabel = None)

plt.xticks(fontsize = 25)

plt.ylabel('Percentage (%)', fontsize = 30)
plt.ylim(bottom = -1, top = 102)
plt.yticks(fontsize = 25)

sns.despine()

vector = means
SI_data = data['SI_Ratio']

```

```

print('Spearman') # Sparman korelasyonu Şekil 4.9.
print('vs SI', spearmanr(data['SI_Ratio'], vector, nan_policy =
'omit'))
print('')

print('Linear Reg.')
print(linregress(SI_data[np.invert(np.isnan(vector))],
vector[np.invert(np.isnan(vector))])

# Lineer regresyon
x = SI_data[np.invert(np.isnan(vector))]
y = vector[np.invert(np.isnan(vector))]

X = sm.add_constant(x)

model = sm.OLS(y, X).fit()

summary = model.summary()

r_squared = model.rsquared
f_statistic = model.fvalue
p_value = model.f_pvalue

print(f"R-squared: {r_squared:.2f}")
print(f"F-statistic: {f_statistic:.2f}")
print(f"P-value: {p_value:.4f}")

print(summary)

slope, intercept, r_val, p_val, _ = linregress(SI_data[np.in-
vert(np.isnan(vector))],
vector[np.in-
vert(np.isnan(vector))])

plt.figure(dpi = 600, figsize = (6,6)) # Saçılım grafiğinin
oluşturulması Şekil 4.9.

scatterplot = sns.scatterplot(x = SI_data.iloc[nai_where], y =
means[nai_where], # hue = 'behav_prof',
color = behav_palette[0],
style = f[nai_where], s = 200,
markers = ['o'], #, 's', 'x'],
edgecolor = 'k', linewidth = 2)
scatterplot = sns.scatterplot(x = SI_data.iloc[res_where], y =
means[res_where], # hue = 'behav_prof',
color = behav_palette[1],

```

```

style = f[res_where], s = 200,
markers = ['s'],#, 's', 'X'],
edgecolor = 'k', linewidth = 2)

scatterplot = sns.scatterplot(x = SI_data.iloc[sus_where], y =
means[sus_where],# hue = 'behav_prof',
color = behav_palette[2],
style = f[sus_where], s = 200,
markers = ['X'],#, 's', 'X'],
edgecolor = 'k', linewidth = 0)

if min(SI_data) < 0:
    x = np.arange(min(SI_data)-1, max(SI_data)+1, 0.1)
    plt.xlim([-1+min(SI_data), max(SI_data) + 1])
else:
    x = np.arange(0, max(SI_data)+1, 0.1)
    plt.xlim([0, max(SI_data) + 1])

plt.plot(x, slope*x+intercept, linestyle = '-', color = 'r', la-
bel = 'Regression line', linewidth = 4)

for axis in ['bottom','left']:
    scatterplot.spines[axis].set_linewidth(1.5)

scatterplot.tick_params(width= 1.5)

scatterplot.text(2.5, 80, 'p = %.3f' %(p_val), fontsize = 25,
fontweight = 450)
scatterplot.text(2.5, 90, 'R2 = %.2f' %(r_squared), fontsize =
25, fontweight = 450)

plt.xlabel('SI Ratio', fontsize = 30)
plt.xticks(ticks = np.arange(1,5,1), fontsize = 25)

plt.ylabel('Percentage (%)', fontsize = 30)
plt.ylim(bottom = -2, top = 102)
plt.yticks(fontsize = 25)
plt.legend(bbox_to_anchor=(1.04, 0.65), loc="upper left", font-
size = 20)
sns.despine()

```

```

# 6. katman için çok renkli sinaps içeren segment oranlarının
gruplar arası karşılaştırması ve sosyal etkileşim oranıyla
orantılanması Şekil 4.16.
# Bu algoritma için her bir grafiğin ayrı düzenlemeye ihtiyaç
duyması sebebiyle herhangi bir fonksiyon oluşturulmadı.
# Algoritma her bir dendritik segment çeşidinin oranların ayrı
ayrı bulunmasını ve
# onların stres duyarlılık fenotipleri arasında karşılaştırılması
ve sosyal etkileşim oranıyla ilişkilendirilmesini gösterir.
# Diğer sinaps türleri değişkenlerin değiştirilmesi yoluyla
incelenebilir.

import statsmodels.api as sm # Kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from scipy import stats
from scipy.stats import mannwhitneyu as mwu
from scipy.stats import kruskal, spearmanr, linregress
import matplotlib as mpl

conn_data = pd.read_pickle('synapses_segments_all_layers.pkl')#
Veri setlerinin yüklenmesi
conn_data = conn_data.sort_values('behav_prof', ignore_index =
True)

data = pd.read_pickle('connection_data+depth_data+segment_cen-
ter_of_mass+synapse_data.pkl')
data = data.sort_values('behav_prof', ignore_index = True)

sus_where = np.where(data['behav_prof'] == 'sus') # Veri setinde
hızlı bulma için kullanılan yardımcı değişkenler
res_where = np.where(data['behav_prof'] == 'res')
nai_where = np.where(data['behav_prof'] == 'nai')

main_path = os.getcwd()
folder_path = os.path.join(main_path, 'main')

black_to_white = sns.color_palette("gray", n_colors=12)
behav_palette = sns.color_palette([black_to_white[-1],
black_to_white[8], black_to_white[0]])

g = np.array(['nai'] * 15, dtype = 'object')

```

```

g[data['behav_prof'] == 'nai'] = 'Nai.'
g[data['behav_prof'] == 'res'] = 'Res.'
g[data['behav_prof'] == 'sus'] = 'Sus.'

LVb_segments = [] #5b katmanı ve 6. katmandaki dendritik segment-
lerin veri setinden çekilmesi
LVI_segments = []

for i in np.arange(len(conn_data)):

    com_x = np.mean(conn_data['dend_x'][i])
    com_y = np.mean(conn_data['dend_y'][i])
    mouse = conn_data['mouse'][i]

    look = look_axis_array[data['mouse'] == mouse][0]
    add_to_com = add_to_col_array[data['mouse'] == mouse][0]
    check_px = np.array(data['check_px'][data['mouse'] ==
mouse])[0]

    if look == 'x':
        com_x = com_x + add_to_com

        if (com_x < check_px[-1]) and (com_x > check_px[-2]):
            LVI_segments.append(conn_data.iloc[i])
        elif (com_x < check_px[-2]) and (com_x > check_px[-3]):
            LVb_segments.append(conn_data.iloc[i])
        else:
            continue

    elif look == 'y':
        com_y = com_y + add_to_com

        if (com_y < check_px[-1]) and (com_y > check_px[-2]):
            LVI_segments.append(conn_data.iloc[i])
        elif (com_y < check_px[-2]) and (com_y > check_px[-3]):
            LVb_segments.append(conn_data.iloc[i])
        else:
            continue

LVb_segments = pd.DataFrame(data = LVb_segments)
LVI_segments = pd.DataFrame(data = LVI_segments)

cyan_ratio_array = []
green_ratio_array = []
yellow_ratio_array = []

```

```

multicolor_ratio_array = [] # Birden fazla sinapsa sahip den-
dritik segment oranının tespit edilmesi

LVI_segments_morethan1 = LVI_segments[LVI_segments['syn_num'] >
1]

for i in np.arange(len(data)):

    mouse = data['mouse'][i]

    LVI_segments_mouse = LVI_segments_morethan1[LVI_seg-
ments_morethan1['mouse'] == mouse]
    seg_color, seg_color_count = np.unique(LVI_seg-
ments_mouse['color'], return_counts = True)
    print(seg_color)
    cyan_ratio = seg_color_count[seg_color == 'cy-
an']/np.sum(seg_color_count)
    if len(cyan_ratio) == 0:
        cyan_ratio = 0

    green_ratio = seg_color_count[seg_color ==
'green']/np.sum(seg_color_count)
    if len(green_ratio) == 0:
        green_ratio = 0

    yellow_ratio = seg_color_count[seg_color == 'yel-
low']/np.sum(seg_color_count)
    if len(yellow_ratio) == 0:
        yellow_ratio = 0

    multicolor_ratio = 1 - (cyan_ratio + green_ratio + yellow_ra-
tio)

    cyan_ratio_array = np.append(cyan_ratio_array, cyan_ratio)
    green_ratio_array = np.append(green_ratio_array, green_ratio)
    yellow_ratio_array = np.append(yellow_ratio_array, yellow_ra-
tio)
    multicolor_ratio_array = np.append(multicolor_ratio_array,
multicolor_ratio)

means = multicolor_ratio_array*100 # Kruskal-Wallis ve MWU test
uygulaması

sus = means[sus_where]
res = means[res_where]
nai = means[nai_where]

```



```

stress = means[stress_where]
print(kruskal(sus,res,nai, nan_policy = 'omit'))
print('sus vs res, ', mwu(sus,res, nan_policy = 'omit'))
print('sus vs nai, ', mwu(sus,nai, nan_policy = 'omit'))
print('res vs nai, ', mwu(res,nai, nan_policy = 'omit'))
print('stress vs nai, ', mwu(stress,nai, nan_policy = 'omit'))

plt.figure(figsize = (4,6), dpi = 600) # Verinin grafiğe dö-
külmesi

boxplot = sns.boxplot(x = g[nai_where], y = means[nai_where],
capwidths = 0.3,
                    color = behav_palette[0],
                    linewidth = 1.5, linecolor = 'k',
                    fliersize = 0, width = 0.45, gap = 2)

boxplot = sns.boxplot(x = g[res_where], y = means[res_where],
capwidths = 0.3,
                    color = behav_palette[1],
                    linewidth = 1.5, linecolor = 'k',
                    fliersize = 0, width = 0.45, gap = 2)

boxplot = sns.boxplot(x = g[sus_where], y = means[sus_where],
capwidths = 0.3,
                    color = behav_palette[2],
                    linewidth = 1.5, linecolor = 'k',
                    fliersize = 0, width = 0.45, gap = 2,
                    medianprops = dict(color = 'w'))

stripplot = sns.swarmplot(x = g, y = means, size = 8.5,
                        edgecolor = 'k',
                        color = 'w', linewidth = 1.5)

for axis in ['bottom', 'left']:
    boxplot.spines[axis].set_linewidth(1.5)

boxplot.tick_params(width= 1.5)

boxplot.set(xlabel = None)
plt.xticks(fontsize = 25)

plt.ylabel('Percentage (%)', fontsize = 30)
plt.ylim(bottom = -1, top = 102)
plt.yticks(fontsize = 25)

sns.despine()

```

```

# 5b katmanı ve 6. katmandaki VH ve BLA sinapslarının uzaysal
incelemesi Şekil 4.20., Şekil 4.21., Şekil 4.22., Şekil 4.23. ve
Şekil 4.24.
# Bu algoritma derin katmandaki VH ve BLA sinapslarına en yakın
ilk 4 BLA ve VH sinapslarının öklid mesafelerini tespit eder.
# İlk aşamada her bir görüntüdeki en yakın sinapsların histogram-
ları çıkarıldı ve oransal sıklıkları gruplar arasında
karşılaştırıldı.
# Sonra rastgele orman sınıflamasıyla gözlemlenen
farklılıklarının rastgele olup olmadığı incelendi.
# Son olarak en yakın ilk 4 sinaps mesafeleri bir matrise dö-
nüştürülerek UMAP ile boyut indirgeme uygulandı.

import statsmodels.api as sm # Kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from scipy import stats
from scipy.stats import mannwhitneyu as mwu
from scipy.stats import kruskal, spearmanr, linregress
import matplotlib as mpl

from sklearn.metrics import pairwise_distances
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score, f1_score
import umap.umap_ as umap
from sklearn.ensemble import RandomForestClassifier

conn_data = pd.read_pickle('synapses_segments_all_layers.pkl') #
Verilerin yüklenmesi

data = pd.read_pickle('connection_data+depth_data+segment_cen-
ter_of_mass+synapse_data.pkl')
data = data.sort_values('behav_prof', ignore_index = True)

conn_data = pd.read_pickle('synapses_segments_all_layers.pkl')
conn_data = conn_data.sort_values('behav_prof', ignore_index =
True)
syn_density_data = pd.read_pickle('normal-
ized_syn_seg_counts.pkl')

sus_where = np.where(data['behav_prof'] == 'sus')
res_where = np.where(data['behav_prof'] == 'res')

```

```

nai_where = np.where(data['behav_prof'] == 'nai')

black_to_white = sns.color_palette("gray", n_colors=12)
behav_palette = sns.color_palette([black_to_white[-1],
black_to_white[8], black_to_white[0]])
stress_palette = sns.color_palette([black_to_white[-1],
black_to_white[3]])

green_yellow_pwise_dist_array_mxn = []
pwise_color_array = []
pwise_behav_array = []
pwise_mouse_array = []
pwise_SI_array = []

reduced_behav_prof = []

for i in np.arange(len(data)):
    x_loc = np.hstack(data['syn_x'][i])
    y_loc = np.hstack(data['syn_y'][i])
    z_loc = np.hstack(data['syn_z'][i])
    colors = np.hstack(data['color'][i])

    if y_loc.max() > x_loc.max():
        deep_where = np.where((y_loc < data['check_px'][i][-1]) &
# 5b katmanı ve 6. katmandaki sinaptik verinin seçilmesi
(y_loc > data['check_px'][i][-3]))
    else:
        deep_where = np.where((x_loc < data['check_px'][i][-1]) &
(x_loc > data['check_px'][i][-3]))

    deep_xloc = x_loc[deep_where]
    deep_yloc = y_loc[deep_where]
    deep_zloc = z_loc[deep_where]
    deep_colors = colors[deep_where]

    deep_locs = np.vstack((deep_xloc, deep_yloc, deep_zloc)).T

    deep_locs_green = deep_locs[deep_colors == 'green']
    deep_locs_yellow = deep_locs[deep_colors == 'yellow']
    deep_locs_cyan = deep_locs[deep_colors == 'cyan']
    deep_locs_green_yellow = deep_locs[(deep_colors ==
'green') | (deep_colors == 'yellow')]

    pwise_color_array.append(Vb_colors[(Vb_colors ==
'green') | (Vb_colors == 'yellow')])
    pwise_behav_array.append(np.array([data['behav_prof'][i]] *
len(Vb_vols_green_yellow), dtype = 'object'))

```

```

    pwise_mouse_array.append(np.array([data['mouse'][i]] *
len(Vb_vols_green_yellow), dtype = 'object'))

    for j in np.arange(len(Vb_locs_green_yellow)): # VH ve BLA
sinapslarına en yakın ilk 4 VH ve BLA sinapslarının seçilmesi
        green_pwise_dist_array = pairwise_dis-
tances(deep_locs_green_yellow[j,:].reshape(1,3),
                                                deep_locs_gre
en)

        green_pwise_dist_array = np.sort(green_pwise_dist_ar-
ray[green_pwise_dist_array>0.01])[:4]

        yellow_pwise_dist_array = pairwise_dis-
tances(deep_locs_green_yellow[j,:].reshape(1,3),
                                                deep_locs_yel
low)

        yellow_pwise_dist_array = np.sort(yellow_pwise_dist_ar-
ray[yellow_pwise_dist_array>0.01])[:4]

        green_yellow_pwise_dist_array =
np.hstack((green_pwise_dist_array, yellow_pwise_dist_array))
        green_yellow_pwise_dist_array_mxn.append(green_yel-
low_pwise_dist_array)

        pwise_SI_array = np.append(pwise_SI_array, data['SI_Ra-
tio'][i])

green_yellow_pwise_dist_array_mxn = np.vstack(green_yel-
low_pwise_dist_array_mxn)
green_yellow_pwise_dist_array_mxn = green_yellow_pwise_dist_ar-
ray_mxn * 0.103
pwise_color_array = np.hstack(pwise_color_array)
pwise_behav_array = np.hstack(pwise_behav_array)
pwise_mouse_array = np.hstack(pwise_mouse_array)

mouse_hists = []

# Sinaps çiftleri ayrı ayrı grafiklendirilmiştir.
# Aşağıda BLA sinapslarına en yakın VH sinapslarının grafik
hazırlığı örneklendirilmiştir.
# Diğer çıktılar değişkenlerin uygun şekilde değiştirilmesiyle
elde edilebilir.

dist_array = green_yellow_pwise_dist_ar-
ray_mxn[:,0][pwise_color_array == 'yellow'] #green_yel-
low_pwise_dist_array_mxn[:,3] en yakın BLA sinapsları için

```

```

mouse_array = pwise_mouse_array[pwise_color_array == 'yellow']#
[pwise_color_array == 'green'] VH sinapsları için

for mouse in data['mouse']:

    delta_mouse = dist_array[mouse_array == mouse]

    counts, bins = np.histogram(delta_mouse, bins =
np.arange(0,260,5)) # 0'dan itibaren 5 µm aralıkla histogram
eldesi

    mouse_hists.append(counts/np.sum(counts))

mouse_hists = np.vstack(mouse_hists)

for i in np.arange(mouse_hists.shape[1]): # Her bir 5 µm'lik his-
togram bandının davranış grupları arasında karşılaştırılması
    if np.sum(mouse_hists, axis = 0)[i] != 0:
        sus = mouse_hists[sus_where,i]
        res = mouse_hists[res_where,i]
        nai = mouse_hists[nai_where,i]

        if (np.sum(sus) != np.sum(res)) and (np.sum(sus) !=
np.sum(nai)) and (np.sum(res) != np.sum(nai)):
            _, p_val = kruskal(sus[0],
                               res[0],
                               nai[0])

            if p_val <= 1:
                print('between: ', bins[i], '-', bins[i+1])
                print(kruskal(sus[0], res[0], nai[0]))
                print('sus vs res', mwu(sus[0], res[0]))
                print('sus vs nai', mwu(sus[0], nai[0]))
                print('res vs nai', mwu(res[0], nai[0]))
                print('---')

                print('nai vs stress', mwu(np.hstack((sus[0],
res[0])), nai[0], nan_policy = 'omit'))

bin_list = [] # Histogram grafiğinin oluşturulması

for i in np.arange(mouse_hists.shape[1]):

    one_bin = '%d - %d' %(bins[i], bins[i+1])
    bin_list = np.append(bin_list, one_bin)

bin_list_dframe = []

```

```

behav_prof_dframe = []

for i in np.arange(len(data)):
    bin_list_dframe.append(bin_list.astype('object'))

for i in np.arange(len(bin_list)):
    behav_prof_dframe.append(np.array(data['behav_prof']))

bin_list_dframe = np.vstack(bin_list_dframe)
behav_prof_dframe = np.vstack(behav_prof_dframe).T

bin_list_dframe = bin_list_dframe[:, :7]
sample = np.array(['35+'] * 15, dtype = 'object')
sample = sample.reshape(len(sample), 1)
bin_list_dframe = np.concatenate((bin_list_dframe, sample), axis = 1)

behav_prof_dframe = behav_prof_dframe[:, :8]

sample = np.sum(mouse_hists[:, 7:], axis = 1)
sample = sample.reshape(len(sample), 1)
mouse_hists = mouse_hists[:, :7]
mouse_hists = np.concatenate((mouse_hists, sample), axis = 1)

dframe = {'bin': [],
          'behav_prof': [],
          'freq': []}

dframe['bin'] = np.hstack(bin_list_dframe)
dframe['behav_prof'] = np.hstack(behav_prof_dframe)
dframe['freq'] = np.hstack(mouse_hists)

dframe = pd.DataFrame(data = dframe)

dframe['behav_prof'][dframe['behav_prof'] == 'nai'] = 'Naive'
dframe['behav_prof'][dframe['behav_prof'] == 'res'] = 'Resilient'
dframe['behav_prof'][dframe['behav_prof'] == 'sus'] = 'Susceptible'

plt.figure(dpi = 600, figsize = (15, 8))
point_plot = sns.pointplot(x="bin", y="freq",
                           hue="behav_prof", palette = 'dark:k',
                           data=dframe,
                           capsizes = 0.15, markersize = 20, dodge
= 0.45,
                           estimator = 'mean', legend = True,
markers = ['o', 's', 'X'], linestyle=['-', '--', ':'],

```

```

        linewidth = 3, bar = ('ci',95))

for i, artist in enumerate(point_plot.lines):
    if i == 0:
        artist.set_markerfacecolor(behav_palette2[0])
        artist.set_linewidth(3)
        artist.set_markeredgewidth(2.25)
        artist.set_zorder(3)
    elif i == 9:
        artist.set_markerfacecolor(behav_palette2[1])
        artist.set_linewidth(3)
        artist.set_markeredgewidth(2.25)
        artist.set_zorder(3)
    elif i == 18:
        artist.set_markerfacecolor(behav_palette2[2])
        artist.set_linewidth(3)
        artist.set_markeredgewidth(0)
        artist.set_zorder(3)

    elif i == 27:
        artist.set_markerfacecolor(behav_palette2[0])
        artist.set_linewidth(3)
    elif i == 28:
        artist.set_markerfacecolor(behav_palette2[1])
        artist.set_linewidth(3)
    elif i == 29:
        artist.set_markerfacecolor(behav_palette2[2])
        artist.set_linewidth(3)
        artist.set_markeredgewidth(0)

for axis in ['bottom','left']:
    point_plot.spines[axis].set_linewidth(1.5)

point_plot.tick_params(width= 1.5)

sns.despine(right = True)

plt.xlabel('Bins ( $\mu\text{m}$ )', fontsize = 30)
plt.xticks(fontsize = 25, rotation = 25)

plt.yticks(ticks = np.arange(0,1.2,0.2), labels =
np.arange(0,120,20), fontsize = 25)
plt.ylabel('Percentage (%)', fontsize = 30)

plt.legend(bbox_to_anchor=(0.95,1.05), fontsize = 25, mark-
erscale = 1, handlelength = 3)

```

```

plt.title('\nDistance to Closest VH synapses from BLA Synapses\n\n', fontsize = 30)

# Rastgele orman sınıflaması deney verisi ve rastgele dağılmış
# veri için uygulama
# Sınıflamanın doğruluk oranı ve F1 skorları her iki grup için de
# hesaplanmıştır.

accuracy_array = []
F1_array = []

num_split = 10

model = RandomForestClassifier(n_estimators = 75, # Deney verisi
                               için rastgele orman sınıflaması
                               random_state = 1)

random_index = np.random.default_rng(seed=1).permutation(
    green_yellow_pwise_dist_array_mxn.shape[0])

green_yellow_pwise_dist_array_mxn_ = green_yellow_pwise_dist_array_mxn[
    random_index]
pwise_behav_array_ = pwise_behav_array[random_index]

kf = KFold(n_splits = num_split, shuffle = False)

i = 0

for train_index, test_index in kf.split(green_yellow_pwise_dist_array_mxn_):
    X_train_fold, X_test_fold = green_yellow_pwise_dist_array_mxn_[
        train_index], green_yellow_pwise_dist_array_mxn_[
        test_index]
    y_train_fold, y_test_fold = pwise_behav_array_[train_index],
    pwise_behav_array_[test_index]

    model.fit(X_train_fold, y_train_fold)

    test_predictions = model.predict(X_test_fold)

    accuracy = accuracy_score(y_test_fold, test_predictions)
    accuracy_array.append(accuracy)

    F1 = f1_score(y_test_fold, test_predictions,
                  labels = np.unique(pwise_behav_array), average
= None)

```



```

F1_array.append(F1)

i = i + 1
print(100*(i / num_split), '% is done')

accuracy_array = np.hstack(accuracy_array)
F1_array = np.vstack(F1_array)

# Rastgele dağılmış veri için rastgele orman sınıflaması
accuracy_array_random = []
F1_array_random = []

num_split = 10

model = RandomForestClassifier(n_estimators = 75,
                              random_state = 1)

kf = KFold(n_splits = num_split, shuffle = False)

i = 0

pwise_behav_array_random = np.random.permutation(pwise_behav_array_.copy())

for train_index, test_index in kf.split(green_yellow_pwise_dist_array_mxn_):
    X_train_fold, X_test_fold = green_yellow_pwise_dist_array_mxn_[train_index], green_yellow_pwise_dist_array_mxn_[test_index]
    y_train_fold, y_test_fold = pwise_behav_array_random[train_index], pwise_behav_array_random[test_index]

    model.fit(X_train_fold, y_train_fold)

    test_predictions = model.predict(X_test_fold)

    accuracy = accuracy_score(y_test_fold, test_predictions)
    accuracy_array_random.append(accuracy)

    F1 = f1_score(y_test_fold, test_predictions,
                 labels = np.unique(pwise_behav_array_random),
average = None)
    F1_array_random.append(F1)

i = i + 1
print(100*(i / num_split), '% is done')

```

```

accuracy_array_random = np.hstack(accuracy_array_random)
F1_array_random = np.vstack(F1_array_random)

# Deneş verisi ve rastgele dađılmış verinin dođruluk oran grafiđi

label_1 = np.array(['Observed']*10, dtype = 'object')
label_2 = np.array(['Random']*10, dtype = 'object')

labels = np.hstack((label_1, label_2))
acc_scores = np.hstack((accuracy_array, accuracy_array_random))

dframe={'labels': labels,
        'accr': acc_scores*100}

dframe = pd.DataFrame(data = dframe)

plt.figure(dpi = 600, figsize = (2,4))
point_plot = sns.pointplot(x="labels", y="accr",
                           hue='labels', palette = behav_palette,
                           data=dframe,
                           estimator = 'mean')
point_plot.set(xlabel=None)

sns.despine(right = True)
plt.ylabel('Accuracy (%)', fontsize = 20)
plt.ylim(top = 102, bottom = 0)
plt.xticks(fontsize = 15, rotation = 45)
plt.yticks(fontsize = 15)

# Deneş verisi ve rastgele dađılmış verinin F1 skor grafiđi
label_1 = np.array(['Observed']*30, dtype = 'object').re-
shape((F1_array.shape))
label_12 = np.array(['Random']*30, dtype = 'object').re-
shape((F1_array.shape))

label_2 = np.array(['Naive']*10, dtype = 'object')
label_22 = np.array(['Resilient']*10, dtype = 'object')
label_23 = np.array(['Susceptible']*10, dtype = 'object')

label_dr = np.hstack(np.concatenate((label_1, label_12), axis =
1))
label_nrs = np.vstack((label_2, label_22, label_23)).T
label_nrs = np.hstack(np.concatenate((label_nrs, label_nrs), axis
= 1))

F1_data = np.hstack(np.concatenate((F1_array, F1_array_random),
axis = 1))

```

```

dframe={'label_dr': label_dr,
        'label_nrs': label_nrs,
        'F1': F1_data}

dframe = pd.DataFrame(data = dframe)

plt.figure(dpi = 600, figsize = (8,6))
point_plot = sns.barplot(x="label_dr", y="F1",
                        hue='label_nrs', palette = behav_palette,
                        data=dframe,
                        estimator = 'mean', errorbar =
('se',1))

point_plot.set(xlabel=None)
sns.despine(right = True)

plt.xticks(fontsize = 25)

plt.yticks(fontsize = 15)
plt.ylabel('F1 Score', fontsize = 25)
plt.ylim(top = 1.02)
plt.legend(bbox_to_anchor=(1.3, 1.00), fontsize = 15)

# UMAP boyut indirgeme ve düşük boyuttaki projeksiyonun grafiğini
oluşturma
model = umap.UMAP(n_components = 2, min_dist = 0.1, random_state
= 42)
scores = model.fit_transform(X)

plt.figure(dpi = 600, figsize = (13,10))
scatterplot = sns.scatterplot(x = scores[:,0], y = scores[:,1],
                             hue = pwise_behav_array, s = 2,
                             palette = sns.color_palette(['#FF9300', '#531B93', '#424242']))

plt.xlabel('UMAP1', fontsize = 30)
plt.ylabel('UMAP2', fontsize = 30)

plt.xticks(fontsize = 25)
plt.yticks(fontsize = 25)

plt.legend(bbox_to_anchor=(1.04, 1.00), loc="upper left", font-
size = 30, markerscale = 10)

```

```
sns.despine()

plt.figure(dpi = 600, figsize = (13,10))
scatterplot = sns.scatterplot(x = scores[:,0], y = scores[:,1],
                              hue = pwise_color_array, s = 2,
                              palette = sns.color_palette(['green', 'gold']))

plt.xlabel('UMAP1', fontsize = 30)
plt.ylabel('UMAP2', fontsize = 30)

plt.xticks(fontsize = 25)
plt.yticks(fontsize = 25)

plt.legend(bbox_to_anchor=(1.04, 1.00), loc="upper left", font-
size = 30, markerscale = 10)

sns.despine()
```

### EK-3: Tez Çalışması ile İlgili Etik Kurul İzinleri



T.C.  
HACETTEPE ÜNİVERSİTESİ  
Hayvan Deneyleeri Yerel Etik Kurulu

Sayı : 52338575 -156

#### HAYVAN DENEYLERİ ETİK KURUL KARARI


TOPLANTI TARİHİ	: 24.12.2019 (SALI)
TOPLANTI SAYISI	: 2019/13
DOSYA KAYIT NUMARASI	: 2019/78
KARAR NUMARASI	: 2019/13-08
ONAY BİTİŞ TARİHİ	: 24.12.2024
ARAŞTIRMA YÜRÜTÜCÜSÜ	: Prof. Dr. Emine Eren KOÇAK
HAYVAN DENEYLERİNDE	
GÖREVLİ ARAŞTIRMACILAR	: Sabahaddin Taha SOLAKOĞLU
DİĞER YARDIMCI	:
ARAŞTIRMACILAR	: Dr. Öğr. Üyesi Şefik Evren ERDENER
ONAYLANAN HAYVAN TÜRÜ ve	: 123 Adet C57BL/6 (9-20 Hafta) ve CD1 Fare (4-6
SAYISI	: Ay)

Üniversitemiz Nörolojik Bilimler ve Psikiyatri Enstitüsü öğretim üyelerinden Prof. Dr. Emine Eren KOÇAK'ın araştırma yürütücüsü olduğu 2019/78 kayıt numaralı "*Medyal Prefrontal Kortekste Sinaptik Bağlantı Reorganizyonu Strese Duyarlılığın Bir Belirleyicisi Olabilir mi?*" isimli çalışma Hayvan Deneyleeri Yerel Etik Kurulu Yönergesi'ne göre uygun bulunarak oy birliği ile onaylanmasına karar verilmiştir.

Araştırma yürütücüsü en geç, onay bitiş tarihinden sonraki 1 ay içerisinde proje sonuç raporunu Kurulumuza teslim etmekle yükümlüdür.

Prof. Dr. Sema ÇALIŞ  
Etik Kurul Başkanı

## EK-4: Tez Çalışması Orijinallik Raporu



### Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author: Sabahaddin Taha Solakoğlu  
 Assignment title: TEZ-FİNAL  
 Submission title: Tez\_15.11.24.pdf  
 File name: Tez\_15.11.24.pdf  
 File size: 6.9M  
 Page count: 202  
 Word count: 39,790  
 Character count: 241,473  
 Submission date: 16-Dec-2024 02:40PM (UTC+0300)  
 Submission ID: 2553626792

T.C.  
 İZMİR EKİTİM ENSTİTÜSÜ  
 SAĞLIK BİLİMLERİ ENSTİTÜSÜ

MEDYAL SPOR ENSTİTÜSÜ KURUMUNUN  
 SINIFLIK BAĞLANTI BEĞERİ GELİŞTİRME VE  
 DUYARLILIK İNİSYATİFİNE İLİŞKİN METİN

Dr. Sabahaddin Taha SOLAKOĞLU




Tıbbi Bilimler Programı  
 DOKÜMAN TEZİ

ANKARA  
 2024

Copyright 2024 Turnitin. All rights reserved.

### Sabahaddin Taha Solakoğlu

#### Tez\_15.11.24.pdf

-  TEZ-FİNAL
-  TAHA-TEZ-ARALIK
-  Sağlık Bilimleri Enstitüsü

#### Belge Ayrıntıları

Gönderi Kimliği  
 trnoid::1:3116545348

Gönderi Tarihi  
 16 Ara 2024 14:39 GMT+3

İndirme Tarihi  
 16 Ara 2024 15:02 GMT+3

Dosya Adı  
 Tez\_15.11.24.pdf

Dosya Boyutu  
 6.9 MB

202 Sayfa  
 39.790 Sözcük  
 241.473 Karakter




## 7% Genel Benzerlik

Her veri tabanı için çıkarılan kaynaklar da dâhil tüm eşleşmelerin kombine toplamı.

### Rapordan Filtrelenen


- Bibliyografya

### Ön Sıradaki Kaynaklar

- 6%  İnternet kaynakları
- 5%  Yayınlar
- 0%  Gönderilen çalışmalar (Öğrenci Makaleleri)

### Bütünlük Bayrakları




#### İnceleme için 1 Bütünlük Bayrağı

-  **Değiştirilen Karakterler**  
20 sayfada 24 şüpheli karakter  
Harfler başka bir alfabeden benzer karakterlerle değiştirilir.

Sistemimizin algoritmaları bir belgede, onu normal bir gönderiden ayırabilecek her türlü tutarsızlığı derinlemesine inceler. Tuhaf bir şey fark edersek incelemeniz için bayrak ekleriz.

Bir Bayrak mutlaka bir sorun olduğunu göstermez. Ancak daha fazla inceleme için dikkatinizi vermenizi öneririz.

### Ön Sıradaki Kaynaklar

- 6%  İnternet kaynakları
- 5%  Yayınlar
- 0%  Gönderilen çalışmalar (Öğrenci Makaleleri)

### Ön Sıradaki Kaynaklar

Gönderi içinde en yüksek eşleşme sayısına sahip kaynaklar. Çıkarılan kaynaklar görüntülenmeyecektir.

1	İnternet	openaccess.hacettepe.edu.tr	1%
2	İnternet	stackoverflow.com	0%
3	İnternet	sciety.org	0%
4	İnternet	www.openaccess.hacettepe.edu.tr:8080	0%
5	İnternet	openaccess.hacettepe.edu.tr:8080	0%
6	İnternet	linuxtut.com	0%
7	İnternet	repository.riteh.uniri.hr	0%
8	Yayın	Sabahaddin Taha Solakoglu, Sefik Evren Erdener, Olga Gliiko, Alp Can, Uygur Sum...	0%
9	İnternet	github.com	0%
10	İnternet	arxiv.org	0%
11	İnternet	lab.hacettepe.edu.tr	0%

## 9. ÖZGEÇMİŞ







