MİMAR SİNAN GÜZEL SANATLAR ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKA ALGORİTMALARININ MİMARİ ŞEMATİK PLAN OLUŞTURMAK İÇİN KULLANIMI

YÜKSEK LİSANS TEZİ

Zeki Mehmet AKÇAN

Enformatik Anabilim Dalı

Bilgisayar Ortamında Sanat ve Tasarım

Tez Danışmanı: Doç. Dr. Bülent Onur TURAN

Ekim, 2022

Zeki Mehmet AKÇAN tarafından hazırlanan tez çalışması YAPAY ZEKA ALGORİTMALARININ MİMARİ ŞEMATİK PLAN OLUŞTURMAK İÇİN KULLANIMI adlı bu tezin Enformatik Anabilim Dalı Bilgisayar Ortamında Sanat ve Tasarım **YÜKSEK LİSANS TEZİ** olarak uygun olduğunu kabul ederim.

Doç. Dr. Bülent Onur TURAN
Tez Yöneticisi

Doç. Dr. Bülent Onur TURAN, Tez Danışmanı
Mimar Sinan Güzel Sanatlar Üniversitesi _____

Prof. Dr. Salih OFLUOĞLU, Üye
Mimar Sinan Güzel Sanatlar Üniversitesi _____

Dr. Öğr. Üyesi Belinda TORUS, Üye
Bahçeşehir Üniversitesi _____

Danışmanım Doç. Dr. Bülent Onur TURAN sorumluluğunda tarafımca hazırlanan YAPAY ZEKA ALGORİTMALARININ MİMARİ ŞEMATİK PLAN OLUŞTURMAK İÇİN KULLANIMI başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Zeki Mehmet AKÇAN

İmza

Eşime,

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, başta bu süreçte bana her zaman destek olan tez danışmanım Dr. Bülent Onur Turan'a, kıymetli zamanını ayırıp sabırla ve büyük bir ilgiyle bana faydalı olabilmek için elinden gelenden fazlasını yapan her çıkmazımda yardımıma koşan Dr. Güven INCE'ye ve ailesinin gösterdiği sabır için kendilerine teşekkürü borç bilirim.

Zeki Mehmet AKÇAN

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

GANs	Generative Adversarial Networks
ÇÜA	Çekişmeli Üretken Ağlar
BDT	Bilgisayar Destekli Tasarım
YSA	Yapay Sinir Ağı
şÇÜA	Şartlı Çekişmeli Üretken Ağlar
ESA	Evrişimli Sinir Ağları
DSA	Derin Sinir Ağları
MDR	Multimedya Belge Tanım
MSE	Ortalama Kare Hatası
RMSE	Kök Ortalama Kare Hatası

ŞEKİL LİSTESİ

TABLO LİSTESİ

YAPAY ZEKA ALGORİTMALARININ MİMARİ ŞEMATİK PLAN OLUŞTURMAK İÇİN KULLANIMI

ÖZET

Günümüzde bir çok sektörde verimi oldukça yüksek yapay zeka teknolojisi ile donatılmış yazılımlar ve robotlar yerini almaya başlamıştır. Yapay zeka teknolojisinin diğer sektörlerde olduğu gibi mimarlık alanında da kullanımı yaygınlaşmıştır. Yüksek işlem hızı ve karar verme aşamasındaki doğruluk payı, yapay zekanın mimarlık alanında yaygınlaşmasını sağlamıştır. Mimarlığın en temel konularından biri mahal planlamasıdır. Çalışmada bu konu üzerinde çözümler oluşturacak algoritmaların uygulanması üzerine veriler üretilmiştir. Çalışma; makine öğreniminin, yapay zeka algoritmalarının, mimari şematik plan tasarımda nasıl kullanılabileceğini, bu tasarımların kentsel ölçekteki etkisini ve bu algoritmaların mimarlık sektöründe nasıl kullanabileceğine dair bilgi üretmeyi amaçlamaktadır. Tez içerisinde aynı muhitte bulunan konut yapılarının mimari planlarını kullanarak, yapay zeka algoritmalarından GANs algoritması ile mimari şematik plan oluşturulmuştur. Çalışmanın iş akışı şu şekildedir:

Literatür taraması ve yapılan araştırmalar sonucunda GANs algoritmaları ve pix2pix ağının tezin amacına daha uygun olabileceği tespit edilmiştir. Pix2pix ağının öğrenebilmesi için daha önce hazırlanmış yeterli miktarda mimari planlar eğitim veri olarak kullanılmıştır. Planlar oda sayısına göre ve bulunduğu kattaki bağımsız bölüm sayısına göre sınıflandırılmıştır. Bu sayede istenilen plan tipine göre öğrenme aşaması özelleştirilmiştir. Planlarda bulunan her bir odanın fonksiyonuna göre ayrı renk verilip böylelikle algoritmanın odaları seçmesi sağlanmıştır. GANs algoritmasının çalışabilmesi için, planlar dış konturlerine kadar siyaha boyanarak üretici (generator) modele girdi (input) olarak , odaların fonksiyonlarına göre renklendirildiği planlar ise ayrıştırıcı (discriminator) modeline gerçek data olarak verilmiştir. Bu sayede farklı gabarilere ve plan çözümlerine göre algoritmanın öğrenme işlemi yapılmıştır. Bunun sonucunda üretken model, daha önce öğrenme datasında bulunmasa da dış konturleri belirlenmiş herhangi bir alan için mimari şematik plan üretme parametrelerine sahip olmuştur.

Anahtar Kelimeler: Çekişmeli Üretken Ağlar, Makine Öğrenimi, Mahal Planlaması, Yapay Zeka, pix2pix

USE OF ARTIFICIAL INTELLIGENCE ALGORITHMS TO CREATE ARCHITECTURAL SCHEMATIC PLAN

ABSTRACT

Today, in many sectors, software and robots equipped with highly efficient artificial intelligence technology have begun to take their place. Artificial intelligence technology has become widespread in the field of architecture, as in other sectors. The fact that the processing speed of algorithms cannot be compared with humans is one of the reasons that will cause artificial intelligence to become widespread in the field of architecture. The main subject of architecture is site planning. In the study, data were produced on the application of algorithms that will create solutions on this subject. The study aims to produce information on how machine learning and artificial intelligence algorithms can be used in architectural schematic plan design, the impact of these designs on an urban scale, and how these algorithms can be used in the architectural sector. In the thesis, an architectural schematic plan was created with the GANs algorithm, one of the artificial intelligence algorithms, by using the architectural plans of the residential buildings in the same neighborhood. The workflow of the study is as follows:

As a result of literature review and research, it has been determined that GANs algorithms and pix2pix network may be more suitable for the purpose of the thesis. In order for the Pix2pix network to learn, 150 previously prepared architectural plans were used as the taught data. The plans are classified according to the number of rooms and the number of independent sections on the floor. In this way, the learning phase is customized according to the desired plan type. Each room in the plans is given a different color according to its function, thus enabling the algorithm to select the rooms. In order for the GANs algorithm to work, the plans in which the outer contours of the plans were painted black were given as noise to the generator model, and the plans in which the rooms were colored according to their functions were given to the discriminator model as real data. In this way, the learning process of the algorithm was performed according to different gauges and plan solutions. As a result, the generator model has the parameters to generate architectural schematic plan for any area whose outer contours have been determined, although it has not been found in the learning data before.

Keywords: Generative Adversarial Networks, Machine Learning, Space Layout, Artificial Intelligence, Pix2Pix

1. GİRİŞ

Mimari plan tasarımı, mimarlığın en temel konusu olarak, geçmişten günümüze aynı iç güdüler doğrultusunda ve yaklaşık olarak aynı yöntemler ile oluşturulmaktadır. Amaç önceleri barınmayı, besin depolamayı ve güvenliği sağlamak olsa da konforu sağlayacak yapılar üretilmiştir. Bu yapıları üretirken, mahal planlamasını o zamanın şartlarına ve konuma göre belirli kısıtlar altında tasarlanmıştır. Bu çalışmada mimarlık tarihinin en eski uğraşlarından biri olan mahal planlamasının tasarımı için mevcut teknoloji ve yapay zeka algoritmaları ile nasıl çözümlenebileceği konusu üzerine araştırmalar yaparak önerilerde bulunulmuştur.

Araştırma ilk olarak yapay zeka algoritmalarının yoğun olarak çalışıldığı alanların incelenmesi ile başlamıştır. Araştırmalara göre yapay zeka, online alışveriş ve reklamcılık sektörü başta olmak üzere, internet arama motorlarında, dijital kişisel asistan programlarında, tercüme programlarında, yüz tanıma sistemleri gibi bir çok alanda kullanılmaktadır.

Bu araştırmanın temas edeceği ilk konulardan biri bilgisayara mimari şematik plan tasarımının nasıl öğretileceğidir. Yapılan araştırmalardan ve incelenen örneklerden bu problemi çözebilecek en uygun algoritmanın Generative Adversarial Networks (GANs) türkçe çevirisi ile Çekişmeli Üretken Ağlar (ÇÜA) olduğu sonucuna varılmıştır. ÇÜA, 2014 yılında NIPS konferansında Ian Goodfellow liderliğindeki bir araştırma ekibi tarafından tanıtılmıştır. Şu sözlerle de genel bir tanımlama yapmıştır: ÇÜA, üretken modelleme problemini çözmek için tasarlanmış bir tür yapay zeka algoritmasıdır. Üretken bir modelin amacı, bir eğitim örnekleri koleksiyonunu incelemek ve bunları oluşturan olasılık dağılımını öğrenmektir (Goodfellow ve ark., 2020).

ÇÜA'lar hakkında Facebook yapay zeka direktörü olan Yann LeCun şu ifadeleri kullanmıştır: *Derin öğrenmede ilginç gelişmeler olmakta... Bana göre bunlardan en önemlisi çekişmeli eğitim veya Çekişmeli Üretken Ağlardır. Önerilmekte olan bu yapı ve çeşitleri, geçtiğimiz on yılın en ilginç fikridir* (Fogg, 2018).

ÇÜA modellerine şematik plan çizimini öğretirken, üretici modelin daha doğru sonuçlar verebilmesi için veri tabanını gruplara ayırma ve renklendirme işlemi yapılmıştır. Mimari planları odaların fonksiyonlarına göre renklendirerek, odaları renklere göre ayırt etmesini sağlamıştır. Planları kattaki daire sayısına göre de gruplandırılmış, katta istenilen daire sayısına göre uzmanlaşmış modeller elde edilmiştir.

Çözülmesi gereken ikinci problem ise yeterli miktarda ve doğrulukta veri tabanıdır. Şematik plan veri tabanı toplanmadan önce, Python dili ile şematik planlar üreterek ÇÜA algoritmalarının istenilen sonucu verip veremeyeceği hızlı bir şekilde test edilmiştir. Üretilen yapay planlar ile başarılı sonuçlar aldıktan sonra gerçek datalar ile çalışmalara devam edilmiştir.

Literatür taraması yapıldığında bu konuda yapılan çalışmalar şu şekildedir:
S.Chaillou 2019 yılında, ÇÜA modellerinin sunduğu olanakları ve bunların ilgili kat planı tasarımları oluşturma yeteneklerini sergilemiştir. Bir dizi başlangıç koşulu ve kısıtlamasını göz önüne alarak konut kat planları tasarlamaya yardımcı olması için GAN modelleri ve Pix2Pix algoritmasını kullanmıştır (Chaillou, 2019).

Araştırma nesnesi olarak genel hastanelerin acil servislerinin fonksiyonel yerleşimi, hiyerarşik tasarım kavramlarını birleştirir ve acil servisler için akıllı bir fonksiyonel yerleşim oluşturma yöntemi önerir. Mimari tasarımda akıllı algoritmaların uygulanmasını keşfetmeyi ve acil servislerinin düzenlerinin üretim problemini çözmek için akıllı bir tasarım yöntemi oluşturmayı amaçlamaktadır (Zhao ve ark., 2021).

Mimari kat planlarını analiz edebilmek ve etiketlemek için otomatik bir sistem sunmaktadır. Odaların konumlarını tespit etmek için önerilen sistemler, verilen kat planlarından hem bölümlere nasıl ayrıldığını hem de ayrılan alanlarda hangi bilgilerin olduğunu anlamaya çalışır. Kamuya açık mimari kat planları veri kümesi üzerinde çalışılmıştır (Ahmed ve ark., 2012).

Literatürde yukarıdaki araştırmalara benzer başka çalışmalar da mevcuttur. Bu araştırma yukarıdaki çalışmalardan pix2pix algoritması ile üretilecek planların farklı kısıtlar altında üretilip birbirleri ile performans karşılaştırması yapılması ile ayrışır.

Bu tez çalışması beş bölüme ayrılmıştır. Birinci bölümünde yöntem , amaç ve araştırma soruları ile ilgili yapılan çalışmalara yer verilmiştir. İkinci bölümünde ya-

pay zeka, derin öğrenme, makine öğrenmesi ve ÇÜA gibi tezin temel konuları ile ilgi aydınlatıcı bilgilere yer verilmiştir. Üçüncü bölümde ise mahal planlamasının algoritmalar tarafından oluşturulması ile ilgili benzer yöntemler hakkında bilgiler yer almaktadır. Dördüncü bölümde hipotezin hangi yöntem ile nasıl savunulduğunun detaylı anlatımı mevcuttur. Beşinci bölüm ise çalışmanın sonucunda elde edilen bilginin ileride yapılacak çalışmalarda nasıl kullanılacağı ile ilgildir. Ayrıca bu bölümde pix2pix modellerinin performansı analiz edilmiştir.

1.1 Tezin Amacı

Bu çalışma, mimari plan şeması oluşturma aşamasında yapay zeka teknolojilerinden faydalanarak, verimli ve hızlı çözümlerin nasıl üretilebileceği konusunda bilgi üretmeyi hedeflemektedir. Bu hedef doğrultusunda standartları sağlayan bölgesel mimariye uygun mimari plan veri tabanı kullanılarak, yapay zeka algoritmaları ile standartlara, bölgesel mimariye , bölgesel ve kişisel ihtiyaçlara uygun mimari şematik planların nasıl üretileceği araştırılmıştır. Bu bağlamda;

- Yapay zeka teknolojilerinin mimari alanda kullanımına yönelik olanakların ve desteğinin tespit edilmesi,
- Binaların yeniden üretimine veya kentsel dönüşüme yönelik çalışmalarda yapay zeka teknolojilerinden nasıl faydalanacağının tespit edilmesi,
- Çalışmanın sonucunda ileriye dönük daha fazla ihtiyacı karşılayabilecek çalışmaların nasıl yapılabileceğinin belirlenmesi ve günümüz mimari plan üretim iş akışına nasıl dahil olabileceği konusunda bilgi üretimi

konularına odaklanılmıştır. Belirtilen konular içerisinde araştırmanın ana sorusu aşağıdaki gibi ortaya çıkmıştır: Yapay zeka algoritmaları ile binaların yeniden kullanım veya kentsel dönüşüm sürecinde şematik plan oluşturma aşamasının çözümünü karşılayabilecek düzeyde bilgi sağlayabilir mi ve nasıl sağlayabilir?

1.2 Kapsam ve Yöntem

Çalışma kapsamında yapay zeka teknolojilerinin mimari plan tasarım probleminin çözüm sürecinde işlevselliği ve kullanılabilirliği incelenmiştir. Bu bağlamda öncelikle mevcut yapay zeka algoritmalarının nasıl ve hangi amaçla kullanıldıklarını tespit etmek üzere elektronik kaynaklar, kitaplar ve dergiler üzerinden literatür taraması yapılmıştır. Yapılan araştırmalar ve incelenen örnekler sonucunda hangi algoritmaların mimari plan tasarımı problemi çözümünde uygun olacağı tespit edilmiştir. Belirlenen algoritmaların arasından Pix2Pix algoritmasının çalışmaya konu edilen problemin çözümü için daha uygun olacağı kanısına varılmıştır ve bu algoritma hakkında araştırmalara yoğunlaşılarak devam edilmiştir. pix2pix görüntüden görüntüye çeviri (Image to Image translation) algoritmasıdır. Görüntüden görüntüye çeviri, bir görüntüyü bir etki alanından diğerine dönüştürme işlemidir; hedef, girdi görüntüsü ile çıktı görüntüsü arasındaki eşlemeyi öğrenmektir. Genellikle sıralı görüntü çiftlerinden oluşan bir eğitim veri tabanı kullanılır. Görüntüden görüntüye çeviri algoritmasını kullanan birçok ÇÜA modeli bulunmaktadır. pix2pix ve CycleGAN bunlara örnektir. Bu çalışmada pix2pix kullanımına sebep en önemli neden model eğitimi için gerekli vei tabanı eşleşmiş görüntü çiftlerinden oluşmaktadır.

Bu çalışmaların yanı sıra makine öğrenimi, derin öğrenme ve ÇÜA algoritmaları ile ilgili kapsamlı araştırma için coursera.org platformunda yer alan kurslara (NG, 2018, 2019; Zhou, 2018) ve literatüre başvurulmuştur. Literatürde araştırmalar yapılırken, aramada en çok kullanılan ingilizce ve türkçe terimler şunlardır; üretken tasarım (generative design), hesaplamalı tasarım (computational design), mekan dizimi (space syntax), ÇÜA algoritmaları. Literatürde geçen ingilizce terimler, türkçe çevirisi yapılırken Türk Dil Kurumu bilgisayar terimleri karşılıklar kılavuzu, Türk Dil Kurumu bilişim terimleri sözlüğü ve yaygın kullanımlar göz önünde bulundurularak çevrilmiştir.

ÇÜA algoritmalarının eğitimi için gerekli veri tabanı, belirli bir bölgenin mevcut ruhsat projelerinden faydalanılmıştır. Aynı yönetmeliklere tabi, benzer emsal hakkına sahip bu planlar, Autodesk Autocad[1] programı kullanılarak şematik dizayn fazına uygun olarak düzenlenip, derlenmiştir. Çalışmanın hedefleri ile ilişkili olarak apartman mimari planları seçilmiştir. Bu planlar katta bulunan daire sayısına ve oda sayısına

[1] https://www.autodesk.com/products/autocad/overview

göre gruplandırılmıştır.

Mimari plan veri tabanı kullanılarak eğitilecek ÇÜA modellerinin yazımı için Pycharm[2] programı kod geliştirme arayüzü olarak kullanılmıştır. Bu program kullanım rahatlığı ve Python[3] dili ile uyumundan dolayı seçilmiştir. Python programlama dilinin makine öğrenimi ve yapay zeka algoritmaları ile ilgili kütüphanelerinin yoğun olması nedeniyle bu dilin seçiminde tercih sebebi olmuştur. Python kütüphanelerinden ağırlıklı olarak makine öğrenimi ve yapay zeka algoritmaları için yazılmış PyTorch[4] ve NumPy[5] kütüphanesinden faydalanılmıştır.

1.3 Kısıtlar

Çalışmanın kapsamı ile bağlantılı olarak, mimari plan veri tabanı aynı yönetmeliklere tabi tutulmuş, çevresel şartlarının aynı olduğu , İstanbul Ataşehir bölgesi için tasarlanmış mimari planlardan elde edilmiştir. Bu veri tabanının kullanılmasının sebebi, öğrenilen verinin araştırma sorusu olarak da belirtilen yeniden kullanım veya kentsel dönüşümde kullanılabilirliğini araştırmaktır. Ayrıca bölgesel şartlara ve standartlara uygun mimari planlar üretmektir. 150 adet plan veri tabanı içerisinde çoğunlukla katta iki daire olmak üzere, tek, üç, dört ve beş bağımsız bölümlü olarak kategorize edilmiştir. İhtiyaçlar doğrultusunda ayrı ayrı modeller eğitilerek kattaki daire sayısı kategorilerine göre uzmanlaşacaktır. Kategorize edilmesi eğitim veri tabanını daha tutarlı yapacak, üretilen planlar istenilen sonuca daha uygun olacaktır.

Ataşehir bölgesi, bölgesel olarak gecekondulaşma ve kentsel dönüşüme uygun olduğundan ve birçok yeni yapılaşmanın bulunması sebebi ile bu çalışma için örnek alınan bölge olmuştur. Kentsel dönüşüme uygun olması, çalışmanın amacında değinildiği gibi üretilen planlar ile bölgesel mimariye uygun daha hızlı verimli mimari planlar üretimi için önemlidir. Yeni yapılaşmanın çok sayıda olması ise yeni standartlara uygun, günümüz ihtiyacına yönelik mimari planlara erişim olmasıdır. Ayrıca, yeni yapılaşmanın sağladığı mimari planlar bilgisayar ortamında istenilen formatta elde edilmesi açısından eski yapıların arşivlenmiş mimari plan kaynaklarına göre daha mümkündür.

[2]https://www.jetbrains.com/pycharm/
[3]https://www.python.org/
[4]https://www.pytorch.org/
[5]https://www.numpy.org/

2. YAPAY ZEKA, YAPAY SİNİR AĞLARI VE DERİN ÖĞRENME

Bu bölümde yapay zeka ve onunla ilişkili kavramlar ile ilgili bilgilendirici metinler ve örnekler oluşturulmuştur. Makine öğrenimi ve yapay zeka teknolojisinin kullanımı; yüksek teknoloji, daha fazla erişilebilirlik ve sunduğu avantajlar ile uyumlu bir şekilde artmaktadır. Nasdaq tarafından yapılan araştırmaya göre farklı firma türlerinden alınan bilgiler doğrultusunda yapay zeka ve makine öğrenimi teknolojilerinden en fazla şu konularda fayda sağlanmıştır:

İyileştirilmiş müşteri deneyimi: Şirketlerin %86'sı yapay zeka yazılımlarının müşteri deneyimlerini iyileştirmelerine olumlu etkilerinin olduğunu bildirdi. Sonuç olarak, müşteri memnuniyeti arttı.

Bilgilendirilmiş karar verme: Lider konumdaki iş insanlarının %75'i için yapay zeka yazılımlarının, karar verme ve stratejinin geliştirilmesine yardımcı olduğunu ve daha iyi sonuçlara ve rekabet avantajlarına yol açtığını bildirmiştir.

Yenilikçi: Herhangi bir yapay zeka teknolojisini tam olarak uygulayan kuruluşların %75'i, halihazırda bulunan ürün ve hizmetler üzerinde yenilik yaptıklarını ve hedef müşterilerinin ihtiyaçlarına daha uygun şekilde tekliflerini iyileştirebildiklerini bildirmiştir.

Maliyetten kazanç: Süreçleri kolaylaştırmak ve gereksiz harcamaları bulmak, yapay zekanın en başarı olduğu işlerdendir. Kuruluşların %70'i, yapay zeka teknolojisinin maliyetleri düşürmelerine ve sermayeden tasarruf etmelerine yardımcı olduğunu bildirmiştir.

Üretkenlik: Kuruluşların yaklaşık üçte ikisi, AI çözümlerini uyguladıktan sonra üretkenlik artışı bildirdi. Yapay zeka, personel için bir destek sistemi olarak insanların işlerini kolaylaştırır ve her zamankinden daha fazlasını yapmalarına yardımcı olur. Ekibinize uygun doğru araçlarla, çalışanların moralini ve elde tutma oranını bile iyileştirebilir(Url-09).

2.1 Yapay Zeka Nedir ?

John McCarthy 2007 yılında yayınladığı makalesinde yapay zekayı ”Akıllı bilgisayar programları yapma bilimi ve mühendisliğidir. İnsan zekasını anlamak için bilgisayar kullanmanın benzer görevi ile ilgilidir, ancak yapay zeka kendisini biyolojik olarak gözlemlenebilir yöntemlere ikna etmek zorunda değildir” sözleri ile tanımlamıştır. (Dick, 2019). Yapay zekayı çalışma şeklinin insan zekasına benzetildiği, insan zekasının dünyada başarması gereken aynı işler ile görevlendirildiği bir bilgisayar programı olarak düşünebiliriz. ”Yapay zeka” kavramının tarihi bilgisayar bilimi kadar eskidir. Fikir babası, ”Makineler düşünebilir mi?” problemini tartışmaya açan İngiliz matematikçi Alan Mathison Turing’dir. 1943’te II. Dünya Savaşı sırasında kripto analizi gereksinimleri için üretilen cihazlar sayesinde bilgisayar bilimi ve yapay zeka kavramları doğmuştur. 1950’lerin sonralarında bir çok araştırmacı yapay zeka konusu ile ilgilenmiş, 1952 yılında ilk ”makine öğrenimi” kavramı doğmuştur bu alandaki ilk program Arthur Samuel tarafından yazılmıştır (Fogg, 2018).

2018 yılında yapılan araştırmaya göre günümüzde yapay zekanın en çok kullanım alanlarından bazıları şu şekildedir(Url-02):

- Doğal dil üretimi: Makinelerin fikirleri doğru bir şekilde iletmesine yardımcı olur,
- Konuşma tanıma veya sesli yanıt: İnsan dilini bilgisayarlar için kullanışlı biçimlere dönüştüren Siri’ye[1] benzer sistemler,
- Görsel temsilci : İnsanlar ile etkileşime girerler, en yaygın örnekleri sesli cevap robotlarıdır,
- Optimize edilmiş donanımlar: Hesaplama zekasına yönelik görev yürütme için tasarlanmış donanımlardır.
- Yüz, ses, parmak izi, iris tanıma : Güvenlik seviyelerini artırmak için geniş bir kullanım alanına sahip sistemlerdir.
- Robotik süreç optimizasyonu: İnsan görevlerini, metin analitiğini, doğal dil üretimini otomatik bir şekilde taklit etmek için komut dosyalarını ve diğer yöntemleri içerir; ayrıca doğal dilin anlaşılmasını kolaylaştırır.

[1] Akıllı asistan servisi

Yapay zeka ile makine öğrenimi ve derin öğrenme konuları birbirleriyle yakın ilişkilidir. Makine öğrenimini yapay zekanın bir alt konusu, derin öğrenme de hem makine öğrenmesinin hem de yapay zekanın bir alt konusu olarak düşünülebilir.

Şekil 2.1 Yapay zeka, makine öğrenimi ve derin öğrenimi bağlantısı (Url-04).

2.2 Makine Öğrenimi

Batta Mahesh 2020 yılında yaptığı çalışmasında makine öğreniminden şu şekilde bahsetmektedir: Makine öğrenimi, bilgisayar sistemlerinin açıkça programlanmadan belirli bir görevi gerçekleştirmek için kullandıkları algoritmaların ve istatistiksel modellerin bilimsel çalışmasıdır. Günlük olarak kullandığımız birçok uygulamada algoritmaları öğrenmek. Google gibi bir web arama motoru internette arama yapmak için her kullanıldığında, bu kadar iyi çalışmanın nedenlerinden biri, web sayfalarının nasıl sıralanacağını öğrenen bir öğrenme algoritmasıdır. Bu algoritmalar veri madenciliği, görüntü işleme, tahmine dayalı analitik vb. gibi çeşitli amaçlar için kullanılır. Makine öğrenimini kullanmanın en büyük avantajı, bir algoritma verilerle ne yapacağını öğrendiğinde, işini otomatik olarak yapabilmesidir (Mahesh, 2020).

Tarihteki ilk makine öğrenim programı Arthur Samuel tarafından 1959 yılında yapılan bir çalışma ile tanıtıldı. Bu çalışmaya göre: Dama oyunu kullanılarak iki makine öğrenme prosedürü ayrıntılı olarak incelenmiştir. Bir bilgisayarın programlanabileceğini doğrulamak için yeterli çalışma yapıldı, böylece programı yazan kişi tarafından oynanabileceğinden daha iyi bir dama oyunu oynamayı öğrenecek. Dahası, sadece oyunun kuralları, yön duygusu ve oyunla bir ilgisi olduğu düşünülen, ancak doğru işaretleri ve göreceli ağırlıkları bilinmeyen ve belirtilmemiş olan parame-

trelerin gereksiz ve eksik bir kısmı verildiğinde, bunu oldukça kısa bir sürede yapmayı öğrenebilir. Bu deneylerle doğrulanan makine öğreniminin ilkeleri elbette diğer birçok durum için de geçerlidir (Samuel, 1959).

2.2.1 Algoritma ve Algoritma Öğrenimi

Algoritma bir problemin çözümü için birbirini takip eden işlemler bütünüdür. Bir probleme çözüm üretmek veya belirlenen amaca ulaşabilmek için tasarlanmış yola ve birbirini takip eden işlem adımlarına algoritma denir. Bilgisayar bilminin temel kavramıdır ve günümüz teknolojisi ile bütün disiplinlere etki etmektedir. Algoritmik düşünce sayesinde, yani amaca giden her bir işlemdeki performansı, artıları ve eksileri düşünme ile tasarımda bütünü oluşturan her bir girdinin değişken özelliklerine göre tasarımın yeniden analiz edilmesi ve üretilmesi mümkündür. Amaç tasarımı oluşturan girdiler arasındaki ilişkinin kurgusunu yapabilmektir (Erbaş, 2015).

Günümüzde, hemen her alanda teknolojik cihazlar kullanılmaktadır. Sahip oldukları algoritmalar, bu teknolojik cihazların etkili ve vazgeçilemez ölçüde tercih edilmesinin ve kullanılmasının ana sebebidir. Üzerinde çalışılmış ve hatasız bir algoritmanın, sonuca ve hedeflenene en kolay ve kısa şekilde ulaşmasından dolayı çok karışık olabilen çok fazla adım kolay bir şekilde istenilen hedefe varmaktadır.

Algoritma öğrenmesi, veri tabanlarını kullanarak öğrenebilme algoritmasıdır (Bengio ve ark., 2017). Öğrenmeden kastedilen ise en popüler tanımı ile ”Bir bilgisayar programının, P ile ölçülen T’deki görevlerdeki performansı deneyim E ile gelişirse, bazı görev sınıfı T ve performans ölçüsü P ile ilgili olarak deneyim E’den öğrendiği söylenir” (Mitchell, 1997) şeklinde tariflenmiştir.

(Nourian ve ark., 2013) makalesi hesaplamalı tasarımı daha iyi anlayabilmek için başarılı bir örnektir. Mimari mahal dizilimlerinin konfigüratif tasarımı için parametrik CAD programı olarak geliştirilen tasarım metodolojisi ve araç takımı tanıtmaktadır. Tasarımcıların, işlevsel alanların birbirine bağlanması için gereken yolu çizerek yerleşim planı sürecini başlatabileceği bir araç seti oluşturmuştur.

2.2.1.1 Algoritma Türleri

Bir programın geliştirilmesinde kullanılacak olan doneler, yapıları ve algoritmaları ile doğrudan yaratılan sisteme uyum göstermektedir. Program yazımı ile ilgilenen kişiler istenilen hedefe varabilmek adına, halihazırda yaratılmış algoritmaları kullanabilir ya da kendi gereksinimlerine göre yeni algoritmalar da yaratabilirler. Gerekli araştırmalar yapılırken; bir sorunla veya çözümlenmesi gereken bir başlıkla daha önceden de karşılaşılıyorsa, bu bahsi geçen sorunun çözümü için çalışılmış ve geliştirilmiş algoritmalar var demektir. Daha önceden geliştirilen algoritmalar kullanılabilir veya kişinin tercihine göre üzerinde çalışılıp geliştirilerek mevcut probleme uyarlanabilir. Algoritmalar; karmaşıklıklarına, uygulama şekillerine, kullanıldıkları alanlarına ve tasarım yöntemlerine göre çeşitli tiplere ayrılmaktadır. Bunların bazılarına örnek vermek gerekirse aşağıdaki gibi örneklendirilebilir;

Arama Algoritması

Dijital sistemlerde arama yapılmak istenildiğinde girilen filtre ve özelliklere göre arama yapılmak üzere kullanılabilecek bir algoritma türüdür. En sık kullanım yeri internette arama yapabilmek için kullanılan arama motorlarıdır.

Sıralama Algoritması

Dağınık halde bulunan verileri belirli özelliklerine göre sıralı hale getirebilmek için kullanılır. Sıralama algoritmalarının kullanıldığı uygulamalarda genellikle isimler alfabetik dizilime, sayılar ise matematiksel büyüklüklere göre sıralanabilirler.

Kriptografik Algoritması

Günümüzde internet alt yapısına bağlı olarak güvenlik terimi ön plana çıkmaktadır. Verilerin güvenli bir şekilde aktarılması ve elde edilmesi için kriptografi algoritmasıyla çeşitli şifreleme ve çözümleme algoritmaları sunulmaktadır. Kriptografi, bir iletinin birden fazla nokta arasında aktarıldığı ortamdan bağımsız bir şekilde güvenli olarak paylaşımını sağlamaktadır. Kriptografi çok geniş uygulama alanlarına dahil olarak günlük hayatın önemli bir parçası olmuştur: Sim kartlar, cep telefonları, uzaktan kumandalar, online bankacılık, uydu alıcıları, vb.

Genetik Algoritma

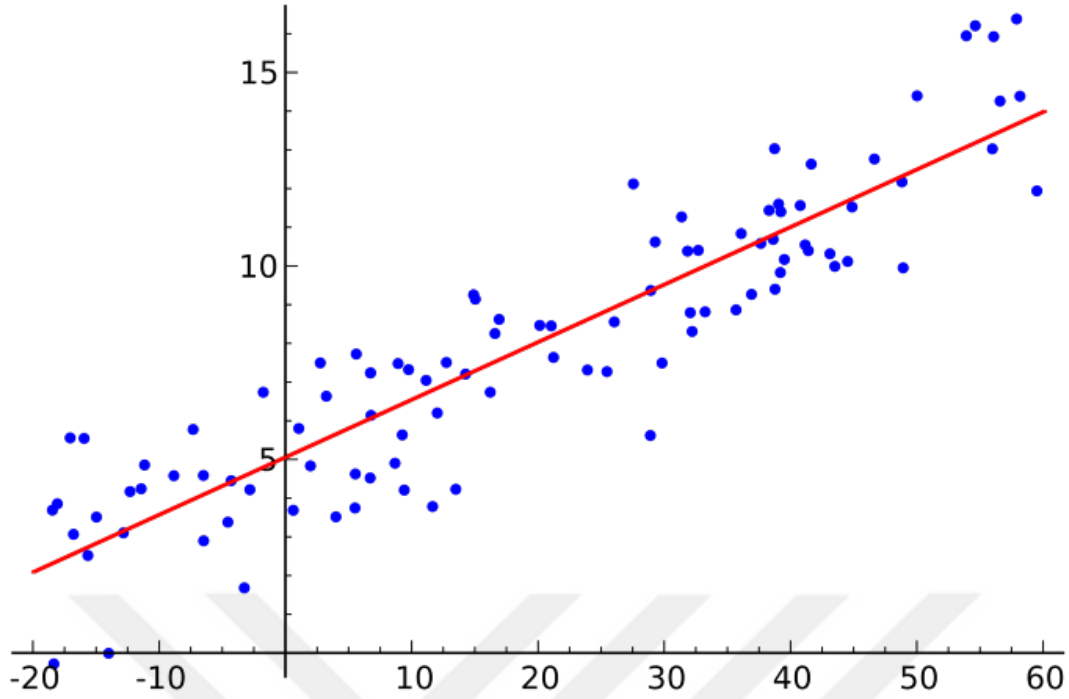
Rassal arama teknikleri kullanılarak çözüm bulunmaya çalışılan, parametre kodlamaya dayanan bir arama algoritma tekniğidir. Genetik algoritmalar optimuma yakın çözümleri mümkün kılmaktadır. Bu algoritmalar parametre setlerinin kodları ile uğraşarak parametrelerle doğrudan ilgilenmemektedirler. Çözüm kümesinin daraltılmış bölgelerinde arama yapmamaktadır. Genetik algoritmaların arama alanları yığının ve popülasyonun tamamıdır. Genetik algoritmanın bir diğer özelliği ise bu algoritmada amaç fonksiyonunun kullanılıyor olmasıdır (Engin, 2001).

2.2.2 Doğrusal Regresyon (Linear Regression)

Doğrusal regresyon bir regresyon problemini çözer. Girdi olarak verilen bir vektör $x \in R^n$ değerlerine göre çıktı olarak bir skaler $y \in R^n$ değerini tahmin edebilen bir sistem oluşturmaktır. Doğrusal regresyon durumunda, çıktı girdinin doğrusal bir işlevidir. (2.1), modelin y'nin üstlenmesini öngördüğü değer olsun. y = ax + b doğrusu için , (2.2) formülüne göre y'nin değerleri tahmin edilir. Buradaki $w \in R^n$ ise vektör parametresidir. Parametreler, sistemin davranışını denetleyen değerlerdir.

$$y = ax + b \tag{2.1}$$

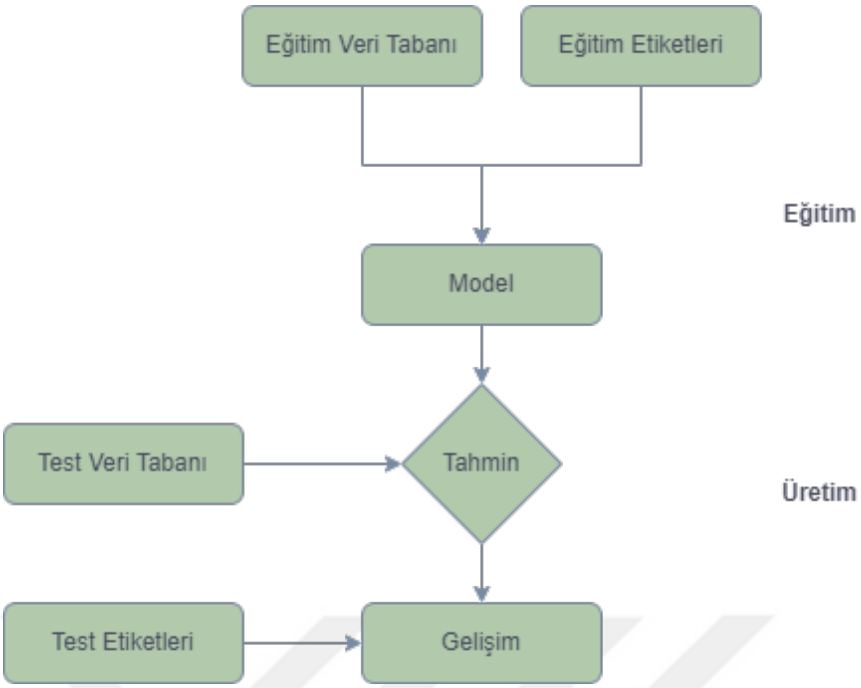
$$\hat{y} = w^T + b \tag{2.2}$$



Şekil 2.2 Doğrusal Regresyon Örneği - from Wikipedia

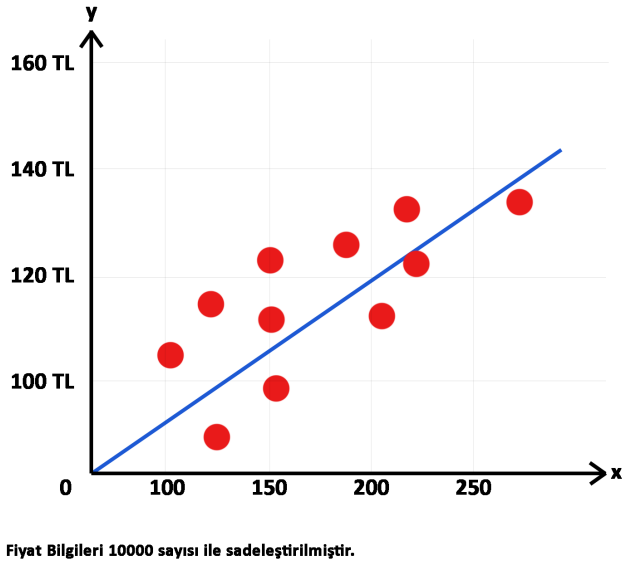
2.2.3 Denetimli Öğrenme Algoritması

İngilizce ”supervised learning alghorithm” olan tekniğin türkçeye denetimli öğrenme algoritması olarak çevrilmiştir. Denetimli öğrenme algoritmaları genel olarak tarif etmek gerekirse, x girdilerinin ve y çıktılarının örneklerinden oluşan bir eğitim seti verildiğinde, bazı girdileri bazı çıktılarla ilişkilendirmeyi öğrenen öğrenme algoritmalarıdır. Çoğu durumda, y çıktılarının otomatik olarak elde etmek zor olabilir ve bir insan ”süpervizör” tarafından sağlanmalıdır, ancak eğitim seti hedefleri otomatik olarak elde edildiğinde bile terim hala geçerlidir (Bengio ve ark., 2017). Denetimli öğrenme problemleri regresyon ve sınıflandırma olarak ikiye ayrılır, bunlar örneklendirilerek açıklanmıştır.

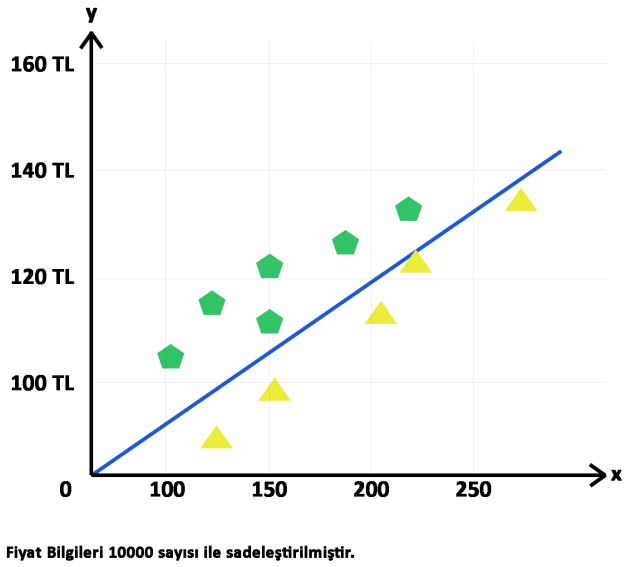


Şekil 2.3 Denetimli Öğrenme Modeli Akış Şeması (Bilgin, 2017)

Gayrimenkullerin sahip olduğu alan özelliklerine göre fiyatlarını tahmin etme algoritmasını incelerken elde bulunan veri tabanında gayrimenkullerin alan bilgileri ve fiyat bilgileri olarak ele alınmıştır. Alan bilgileri x girdisi olarak, fiyat bilgilerini ise y çıktısı olarak tarif edilmiştir. Bu bilgilere sahip bir denetimli öğrenme algoritması fiyat ve alan arasındaki ilişkiyi öğrenerek sadece alan bilgisi bulunan bir gayrimenkul için fiyat tahmininde bulunabilecektir. Bu bir regresyon problemidir. Tahmin edilen fiyatın üstünde veya altında satıldığı ise bir sınıflandırma problemidir.



Şekil 2.4 Regresyon Grafiği



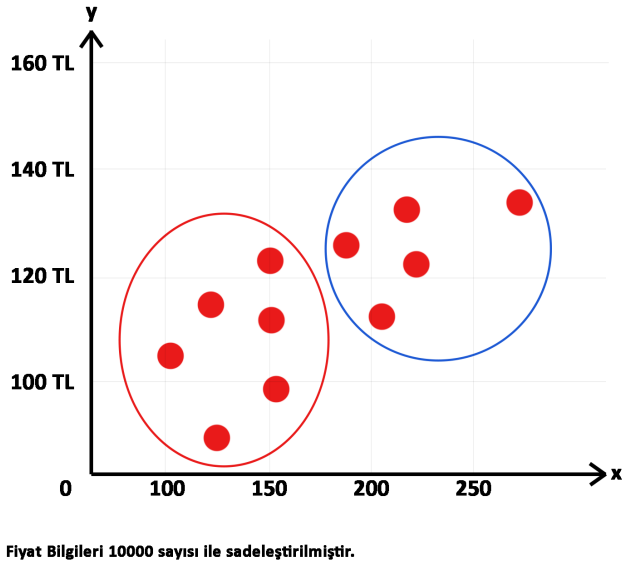
Şekil 2.5 Sınıflandırma Grafiği

2.2.4 Denetimsiz Öğrenme Algoritması

İngilizce ”unsupervised learning alghorithm” olan tekniğin türkçeye denetimsiz öğrenme algoritması olarak çevrilmiştir. Denetimsiz algoritmalar, yalnızca ”özellikleri” deneyimleyen ancak bir denetim sinyali olmayan algoritmalardır. Denetimli ve denetimsiz algoritmalar arasındaki ayrım kesin bir şekilde tanımlanmamıştır, çünkü bir değerin bir süpervizör tarafından sağlanan bir özellik mi yoksa hedef mi olduğunu ayırt etmek için nesnel bir test yoktur. Yalın bir ifade ile, denetimsiz öğrenme, örnek-

lere açıklama eklemek ve çıktıların değerlerini belirlemek için insan emeği gerektirmeyen bir dağıtımdan bilgi çıkarma girişimlerinin çoğunu ifade eder. Bu terim genellikle yoğunluk tahmini, bir dağılımdan örnekler çizmeyi öğrenme, bazı dağılımlardan veri çıkarmayı öğrenme, verilerin yakınında bulunduğu bir benzerlik bulma veya verileri ilgili örneklerden oluşan gruplar halinde kümeleme ile ilişkilidir (Bengio ve ark., 2017).

2.2.3 Denetimli Öğrenme bölümünde verilen örnekten devam ederek, denetimsiz öğrenme algoritmaları hangi gayrimenkul dairelerin benzer özellik gösterdiğini, benzer konumlarda yer aldığını veya benzer malzeme kalitesi kullanıldığını tespit edebilir ve bu benzer daireleri gruplandırır.



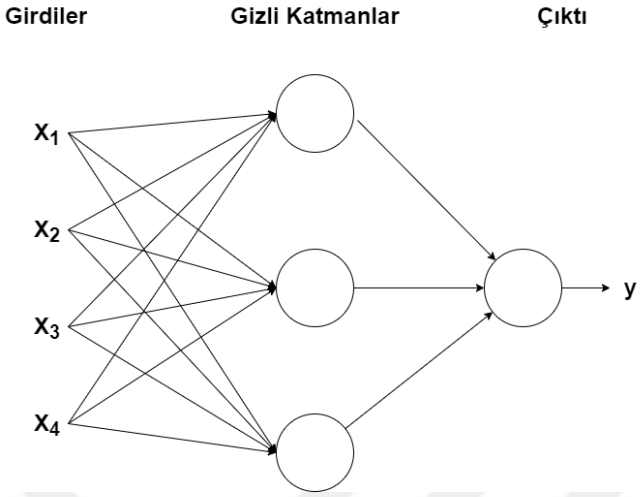
Şekil 2.6 Denetimsiz Öğrenme Gruplandırma Grafiği

2.3 Yapay Sinir Ağları

Yapay sinir ağları (YSA), birbirine bağlı çok sayıda basit işlemciye sahip kitlesel olarak paralel ve çalışma şekli olarak insan beyni ile benzer sistemlerdir. Biyolojik sistemlerin çalışmasında yapılan gözlemlerden esinlenerek ortaya çıkmıştır. İnsan beyni gibi öğrenme yetisine sahip mevcut verileri kullanarak yardım almadan bilgi üretebilen ve keşfedebilen sistemlerdir (Akdoğan, 2017; Jain ve ark., 1996; Priddy & Keller, 2005).

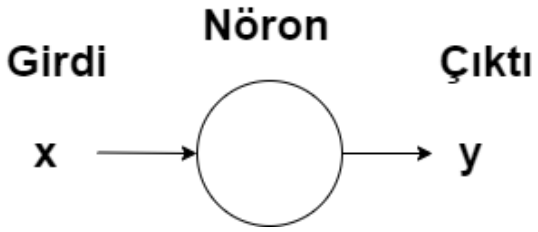
Şekil 2.7'da YSA modeli gözükmektedir. Bu modele girdi (input) olarak ver-

ilen verisetleri girdi katmanında yer almaktadır. Buna göre girdi olarak verilenler x_1, x_2, x_3, x_4 olarak adlandırılmıştır.



Şekil 2.7 Standart Yapay Sinir Ağları Modeli

Nöronlar (Şekil 2.8) verilerin analiz ve hesaplandığı kısımdır. Analiz aşamasından geçerek güncellenen veriler yine nöron yapılarında depolanmaktadır. Nöronların bulunduğu, girdilerin hesaplanıp analiz edildiği kısım gizli katmanlar (hidden layers) olarak adlandırılır. Hesaplamalar ve analiz sonucu çıkan sonuç ise çıktı (output) katmanında bulunur.



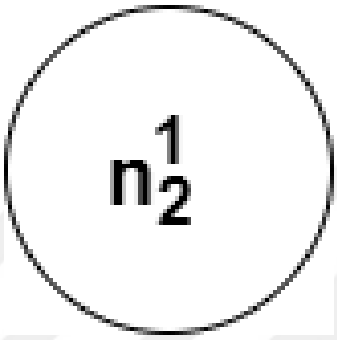
Şekil 2.8 Nöron

Örnek bir model ile YSA'ların nasıl çalıştığını daha detaylı irdelenecektir. Örnek YSA verilen özelliklere göre gayrimenkul fiyatlarını tahmin eden bir model olsun. Elimizde gayrimenkuller ile ilgili detaylı bilgilerin olduğu veri tabanı olsun. Bu veri tabanına göre, bir gayrimenkulun fiyatını ve şu özellikler bilinmektedir.

- x_1 = Alan bilgisi
- x_2 = Kullanılan malzemelerin kalitesi

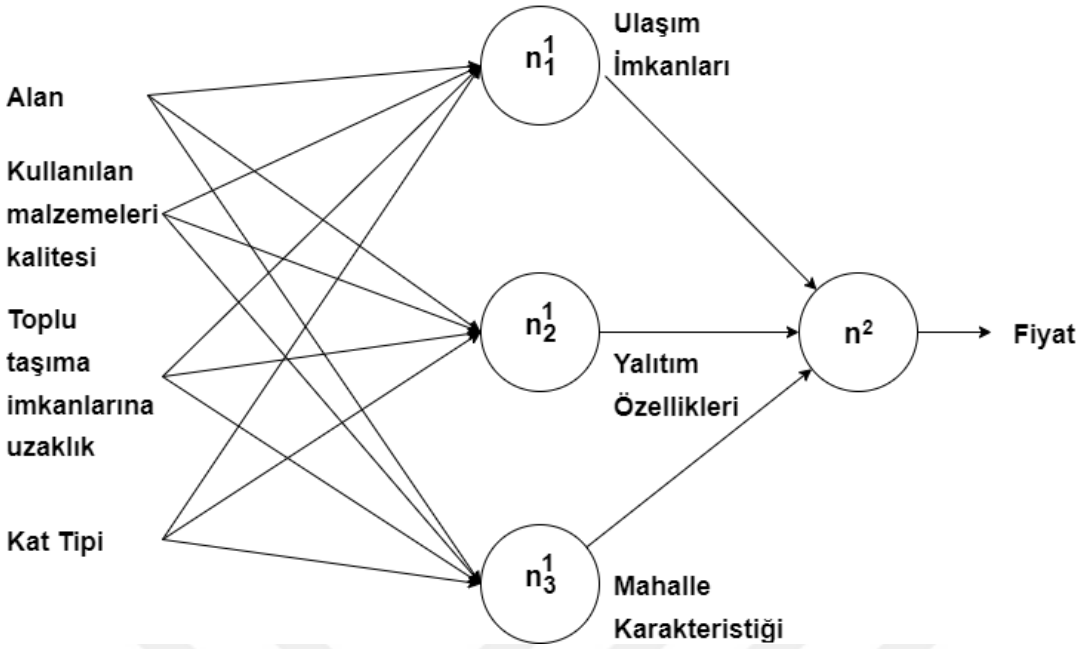
- x_3 = Toplu taşıma imkanlarına uzaklık
- x_4 = Kat Tipi

Bu verilenlere göre y'yi yani evin fiyatını bulmaya çalışacaktır. Şekil 2.7'da olduğu gibi üç nöronlu gizli katman oluşturulmuştur. Bir YSA içerisinde nöronlar sıralarına göre numaralandırılır. Bu numaralandırılmada n harfi nöronu temsil eder, n harfinin sağ alt köşesine nöron sırası, sağ üst köşesine ise katman sırası yazılır (Şekil2.9).



Şekil 2.9 Nöron Numaralandırma

Verilen örnekte her bir nöron eğitim veri tabanına göre gayrimenkulun farklı bir özelliği ile ilgili hesaplamalar yapar. Girdi katmanındaki verileri kullanarak n_1^1 nöronu gayrimenkulun *ulaşım imkanları* ile ilgili bilgi verir. Nöron bu hesaplamaları yaparken bu bilgiye en çok etki eden *toplu taşıma imkanlarına uzaklık* verisi olacaktır. *Alan*, *kat tipi* ve *kullanılan malzemelerin kalitesi* verileri gayrimenkulun *ulaşım imkanları* özelliklerine etki etmediğini n_1^1 nöronu eğitim aşamasında öğrenecek parametrelerini de buna göre eğitecektir. n_2^1 nöronu ise *kat tipi, alan* ve *kullanılan malzemelerin kalitesi* verilerinden daha çok etkilenerek *yalıtım özellikleri* ile ilgili hesaplamaları yapacaktır. Bu nöronda ise *toplu taşıma imkanlarına uzaklık* verisinin hesaplamlarda etkisiz kalması beklenecektir. n_3^1 nöronu da mevcut girdi verilerine göre *mahalle karakteristiği* özelliği ile ilgili hesaplamaları yapacaktır. Elde edilen hesaplamalar daha sonra fiyat özelliğini bulmak için n^2 nöronuna girdi olarak verilerek hesaplaması yapılacaktır (Şekil 2.10).



Şekil 2.10 Gayrimenkul Fiyatını Tahmin Eden Yapay Sinir Ağı

Bu konu hakkında daha ayrıntılı ve derinlemesine bilginin verildiği (NG, 2019) çalışması yapay sinir ağlarının detayları ile anlatıldığı başarılı bir örnektir.

2.3.1 Evrişimli Sinir Ağları

İngilizce Convolutional Neural Network ve kısaltması CNN olarak geçmektedir. Öğrenme algoritmaları ile kendini optimize eden nöronlardan oluşmaları bakımından standart YSA'lar ile benzerlik gösterirler. Evrişimli Sinir Ağları (ESA) ve standart YSA arasındaki en önemli fark, ESA öncelikle görüntüler içinde örüntü tanıma alanında kullanılmasıdır. Sinir ağını görüntü odaklı görevler için daha uygun hale getirirken, modeli kurmak için gereken parametreleri daha da azaltır (Albawi ve ark., 2017).

Örnek olarak Imdat As ve ekibinin çalışması gösterilebilir (As ve ark., 2018). Bu çalışmada, belirlenen hedeflere yönelik tasarımın temel yapı taşlarını oluşturan topolojik özelliklerini keşfedip, kullanıcı gereksinimlerine göre yeni tasarımlar oluşturmuşlardır. Bu tasarımları oluştururken ESA ve GAN algoritma tekniklerinden faydalanmışlardır.

2.3.2 Derin Sinir Ağları

Derin sinir ağları (DSA), giriş ve çıkış katmanları arasında gizli katmanlara sahip bir YSA'dır. En basit haliyle, belirli bir düzeyde karmaşıklığa sahip bir sinir ağı, derin sinir ağ olarak nitelendirilir. Bu ağlar, karmaşık matematik modellemesi ile verileri karmaşık yollarla işler.

2.4 Derin Öğrenme

Bilgisayarların deneyimlerden öğrenmesine ve her bir kavramın daha basit kavramlarla olan ilişkisine göre tanımlanmış kavramlar hiyerarşisi açısından dünyayı anlamasına izin vermektir. Bu yaklaşım, deneyimden bilgi toplayarak, insan operatörlerin bilgisayarın ihtiyaç duyduğu tüm bilgileri belirtme ihtiyacını ortadan kaldırır. Kavramların hiyerarşisi, bilgisayarın karmaşık kavramları daha basit olanlardan oluşturarak öğrenmesini sağlar. Bu yaklaşım derin öğrenme olarak adlandırılmaktadır (Goodfellow ve ark., 2016).

2.4.1 İleri Yayılım (Forward Propagation)

Yapay sinir ağları, her birinde özel hesaplamaların yapıldığı nöronlar ve bu nöronların gruplaştığı farklı katmanlardan oluşur. Katmanları; girdi katmanı, gizli katman ve çıktı katmanı olmak üzere 3 gruba ayırabiliriz. Girdi katmanı görsel, işitsel ya da yazın şeklinde olabilir. Sinir ağının yapacağı işlemler verilerin türlerine göre farklılık gösterebilir. Girdi verisi, bir görsel ise, çalışılan işlem renklerin birbirinden ayırt edilmesi ya da objelerin yanal yüzeylerinin ayırt edilmesi olabilir.

Yapay bir sinir ağında katmanlar arasında, her nöronu diğer katmandaki bir nörona bağlayan uzantılar ve bu uzantıların sayısal değerleri vardır. Bu değerlere ağırlık denir. Bu ağırlık değerleri eğitim neticesinde her bir nöronun, çıktı değer için önemini belirtir. Her bir nöronda hesaplanan çıkış değeri, ağırlığı ile çarpılarak diğer katmandaki nöronun girdi değeri hesaplanır. Girdi katmanından çıktı katmanına doğru ilerleyen bu hesaplama akışına ileri yayılım algoritması denir (Tüfekçi & Karpat, 2019).

İleri yayılım aşamasında; yapay sinir ağının o andaki durumunda yapay sinir ağına uygulanan giriş sinyallerine karşı yapay sinir ağının çıkışlarında oluşan değerler bu-

lunur. Geri yayılım aşamasında, çıkışlarda oluşan yanlışlıklar baz alınarak, üzerinde çalışılan devredeki ağırlıkların tekrar düzenlenmesi yapılmaktadır (Ataseven, 2013).

2.4.2 Geri Yayılım (Back Propagation)

Geri yayılım, model eğitimi için geriye doğru hesaplama algoritmasıdır. Çıktı katmanından başlayarak ilk gizli katmana giderek her katman için gradyanları verir. Geri Yayılım algoritması, delta kuralı ve gradyan iniş tekniği kullanarak hata fonksiyonunun minimum değerini arar. Hata fonksiyonunu en aza indiren ağırlıkların öğrenme problemine bir çözüm olduğu düşünülmektedir (Url-10).

2.5 Bilgisayarla Görü (Computer Vision)

Bilgisayarla görü, görüntüleri işlemenin çok çeşitli yollarını ve uygulamaları kapsayan geniş bir alandır. Bilgisayarla görü uygulamaları, yüzleri tanımak gibi insani görsel yetenekleri yeniden üretmekten, tamamen yeni görsel yetenek kategorileri oluşturmaya kadar uzanır. Bilgisayarla görme üzerine yapılan çoğu derin öğrenme araştırması, insan yeteneklerini kopyalamaya odaklanmıştır. Bilgisayarla görü için çoğu derin öğrenme, nesne tanıma veya bazı biçimlerin algılanması için kullanılır; bu, bir görüntüde hangi nesnenin bulunduğunu bildirmek, her nesnenin etrafında sınırlayıcı kutularla bir görüntüye açıklama eklemek, bir görüntüden bir sembol dizisini kopyalamak veya bir görüntüdeki her pikseli ait olduğu nesnenin kimliğiyle etiketlemek anlamına gelir **(Bengio ve ark., 2017)**.

Bilgisayarla görü; nesnelerin yapısı, şekli ve bulunduğu yer vb. olguları elde edilebilmektedir. Elde edilen bu olgular ile belirlenmiş komutların sınırları içinde, sayısal sistemlerin rasyonel seçimler yapabilmesi sağlanabilmektedir. Bilgisayarla görü; robota, kameradan elde edilen görüntüyü işleyerek gerekli bilgileri transfer etmekte ve böylelikle robotun çalışması ile elde edilen görüntülerden, belirlenen sınırlara göre bir netice ortaya koyabilmektedir.

Bu sistem dahilinde sürecin işleyişi; uygulama işleyişinin bir algılayıcının yardımı ile görüntülerin elde edilmesi, işlenmesi, sistem tarafından yorumlanması ve geri dönüş yapması şeklinde ilerlemektedir. Yukarıda tanımlanan tüm bu sistem; bir görseli yükleme, kopyasını oluşturma, renkli görselden gri tonlu bir görsel oluşturma, görsel

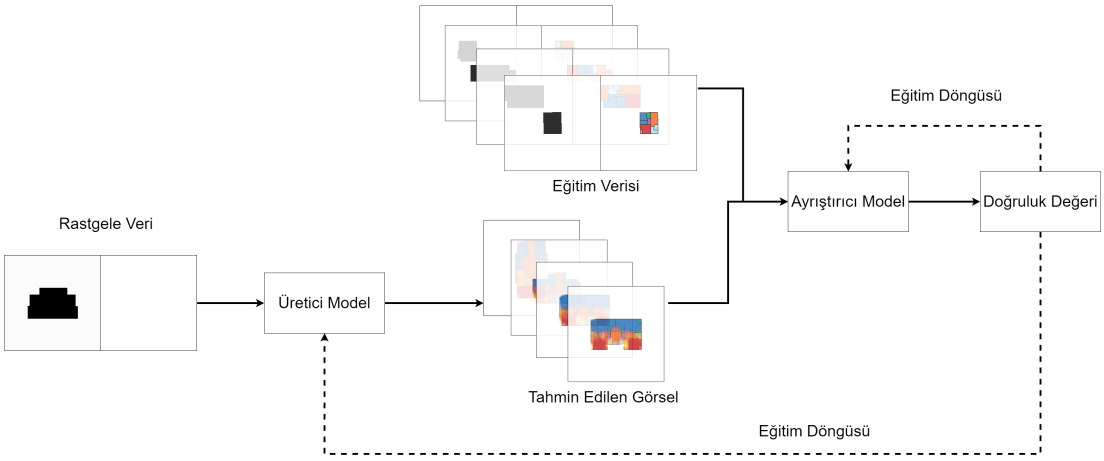
nesnelerin hatlarını çizme, görseli boyama vb. gibi ana resim işleme yöntemlerini bünyesinde barındırır.

Bilgisayarla görü sistemleri işledikleri doneler sebebiyle gözetmen yada hakem gibi denetleyicilerin iş yükünü azaltmaktadır. Bu sistem ile yapılan çalışmalar çerçevesinde de satranç müsabakalarında görevli olan hakemlerin, karar verme yetilerine yardımcı olmayı hedefleyen bir bilgisayarla görü sistemi meydana getirilmiştir. Tüm bu sistem çalışırken, maliyeti düşük, insan gücü ihtiyacını en az seviyeye indiren, müsabaka temsilcilerini rahatsız etmeyecek bir sistem oluşturulmuştur. Buna ek olarak; yaratılan arayüzlerle gerçekleşecek tüm müsabakaları eş zamanlı olarak, isteyen herkesin uzaktan takip edebilmelerine de olanak sağlanabilmektedir (Koray, 2016).

Son yıllarda yapılan çalışmalar arasında; bilgisayarla görü sistemlerine örnek, insansız araç çalışmaları hızla devam etmekte ve dikkatleri üzerine çekmektedir.

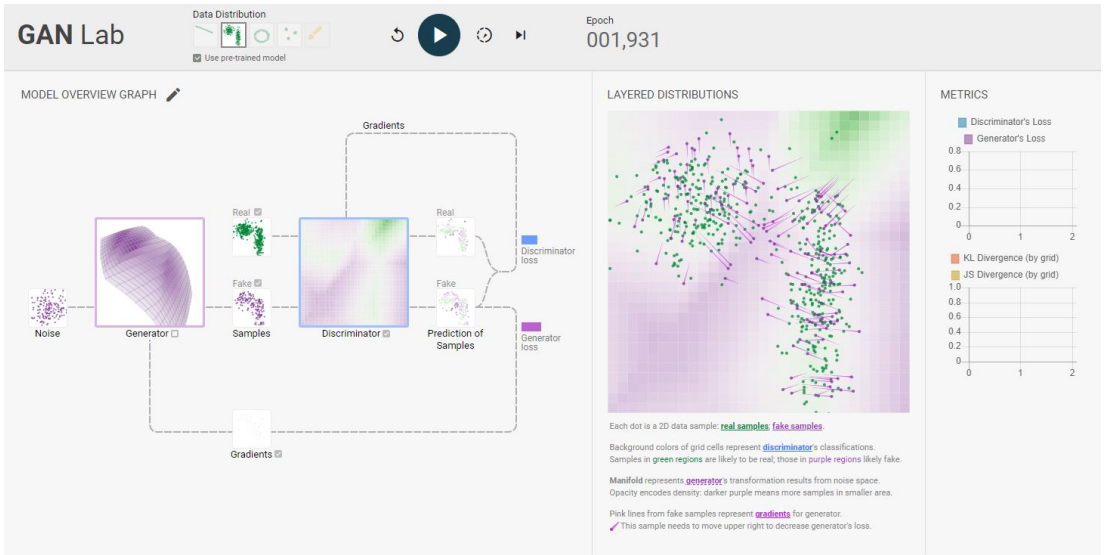
2.6 Çekişmeli Üretken Ağ Algoritmaları

Çekişmeli üretken ağlar, görevi üretici modelleme problemini çözmek olan bir tür yapay zeka algoritmasıdır. Bir üretken modelin amacı, bir eğitim veri tabanındaki örneklerini incelemek ve bunları oluşturan olasılık dağılımını öğrenmektir. ÇÜA daha sonra tahmini olasılık dağılımları ile daha fazla benzersiz örnekler üretebilir. Derin öğrenmeye dayalı bir çok üretken model bulunmaktadır, ÇÜA en başarılı üretken modeller arasındadır. En başarılı ve en çok kullanılan yüksek çözünürlüklü görüntü üretme işlemleridir (Goodfellow ve ark., 2020). ÇÜA'ların amacı yapay görüntüler oluşturmaktır. Bu algoritmalar üretken model ve ayrıştırıcı modelden oluşur. Bu modeller veri tabanındaki görüntüler ile eğitilir. Üretken modele başlangıçta bir girdi verilerek gerçek görüntüye benzer sahte görüntü elde etmesi beklenir. Ayırıştırıcı model ise bu görüntünün sahte olup olmadığını belirlemeye çalışır. Ayrıştırıcı model görüntülerin sahte veya gerçek olduğuna karar verirken kendi parametrelerini yenileyerek geliştirir, üretken model de ayrıştırıcı modelden aldığı gerçeklik değerleri ile daha gerçekçi görüntüler elde etmek için kendini parametrelerini güncelleyerek geliştirir. Bu döngü ile üretken model gerçeğe daha yakın sahte görüntü üretirken aynı zamanda ayrıştırıcı model ise bu görüntüleri daha iyi ayırt etmeye çalışır bkz. Şekil 2.11.



Şekil 2.11 ÇÜA Eğitimi İş Akış Şeması (Kahng ve ark., 2018)

Bruno Gavranović'in 2018 yılında yaptığı çalışmaya göre haftalık olarak ÇÜA makaleleri yayınlanmaktadır. 2014 ve 2018 arasında yaklaşık 500 ÇÜA algoritması yayınlanmıştır (Hindupur, 2018). ÇÜA algoritmalarının çalışma mantığına yönelik hazırlanan bu çalışmada görsel ve uygulamaları olarak konu hakkında detaylı bilgi verilmiştir (Kahng ve ark., 2018). Çalışma içerisinde hazırlanan web tabanlı uygulamada ÇÜA algoritmalarının nasıl çalıştığı simülasyonlar ve interaktif uygulama üzerinden daha anlaşılır hale getirilmiştir (Kahng ve ark., 2019).



Şekil 2.12 GAN Lab web uygulamasından alıntıdır (Kahng ve ark., 2019).

ÇÜA yayınlandığından bu zamana kadar çeşitli araştırmacılar tarafından yüzlerce ÇÜA uzantısı algoritmalar üretilmiştir. Bu çalışmada ÇÜA algoritmalarından DCGAN, CycleGAN, pix2pix ve Conditional GANs algoritmaları detaylı bir şekilde in-

celenmiş birbileri ile kıyaslama yapılarak çalışmanın amacına uygun olan algoritma seçilmştir. Araştırmalar ve incelemeler sonucunda pix2pix algoritması ile devam edilerek testler yapılmıştır.

2.6.1 pix2pix

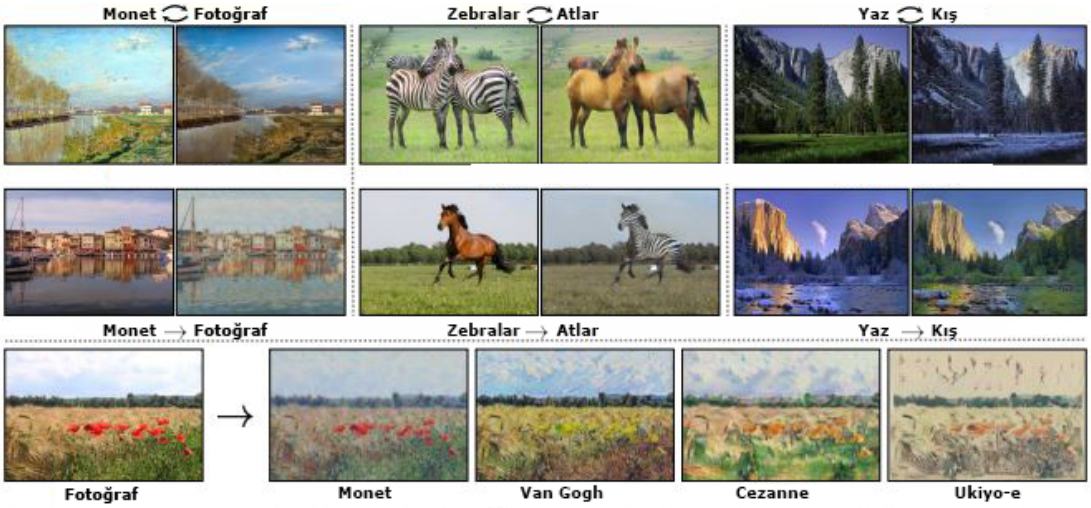
pix2pix görüntüden görüntüye dönüşüm yapabilen ESA ile oluşturulmuş bir modeldir. Bu modelin en önemli avatajı ise diğer ÇÜA modellerine nazaran daha yüksek çözünürlükte görüntüler oluşturabilmesidir. 2017 yılında Philip Isola ve ekibi tarafından geliştirilmiştir. pix2pix modeli etiket haritalarından fotoğrafları sentezlemede, kenar haritalarından nesneleri yeniden yapılandırmada ve görüntüleri renklendirmede etkili olmaktadır. pix2pix modelinin yayınlanmasından bu yana, çok sayıda araştırmacı ve sanatçı, kendi sanatsal deneyleri için pix2pix kullanmıştır (Isola ve ark., 2017). İş akışı için şekil 2.11'ta bulunmaktadır.

Şekil 2.13 pix2pix Örnek (Isola ve ark., 2017)

2.6.2 CycleGAN

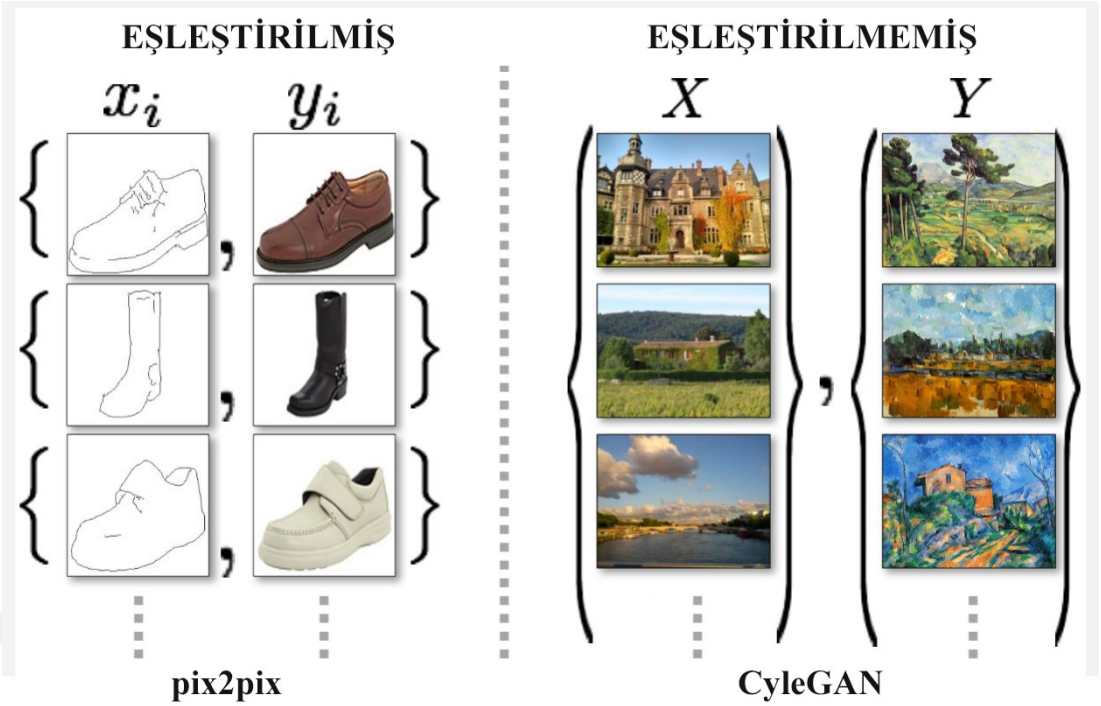
GAN modellerinde hedef, hizalanmış görüntü çiftlerinden oluşan bir eğitim seti kullanarak bir girdi görüntüsü ile çıktı görüntüsü arasındaki eşlemeyi öğrenmektir. Eşleştirilmiş eğitim verilerileri için X bir görüntüyü Y görüntüsüne çevirmeyi öğrenmek için bir yaklaşım sunulmaktadır.

Şekil 2.14'de herhangi iki resim koleksiyonu verildiğinde, CycleGAN algoritması bir resmi, birinden diğerine otomatik olarak çevirmeyi öğrenir ve bunun tersi de geçerlidir. Solda Monet resimleri ve manzara fotoğrafları; ortada zebralar ve atlar; sağda yaz ve kış Yosemite fotoğrafları. Örnek uygulama altta: ünlü sanatçıların tablolarından oluşan bir veri tabanını kullanarak, algoritmaya girdi olarak veilen fotoğrafları ilgili stillere dönüştürmeyi öğrenir (Zhu ve ark., 2017).



Şekil 2.14 CycleGAN Örnekleri (Zhu ve ark., 2017).

CycleGAN algoritmasını pix2pix algoritmasından ayıran en temel fark kullanılan görseller eşleşmemiş görsellerdir. Örnek olarak bu çalışmada kullanılan mimari kat planlarının eşleği olarak aynı plan ile aynı geometriye sahip planın dış kontürlerinden itibaren içlerinin siyah renk ile doldurulduğu görseller bulunmaktadır. Bu siyah renk ile boyanmış görseller (girdi görselleri) ile hedef görüntüler plan sınırları bağlamında birebir eşleştirilmiştir. Bu çalışma CycleGAN algoritması ile yapılsaydı kullanılan görsellerin eşleştirmesine gerek kalmadan iki ayrı grup halinde mimari kat planları ve içleri siyah renk ile kaplı kat planları şeklinde görseller kullanılabilirdi. Bu iki grup içerisinde herhangi bir eşleşmeye gerek duymadan eğitim yapılabilir.



Şekil 2.15 CycleGAN ve pix2pix örnek veri tabanı görselleri (Fordjour, 2019).

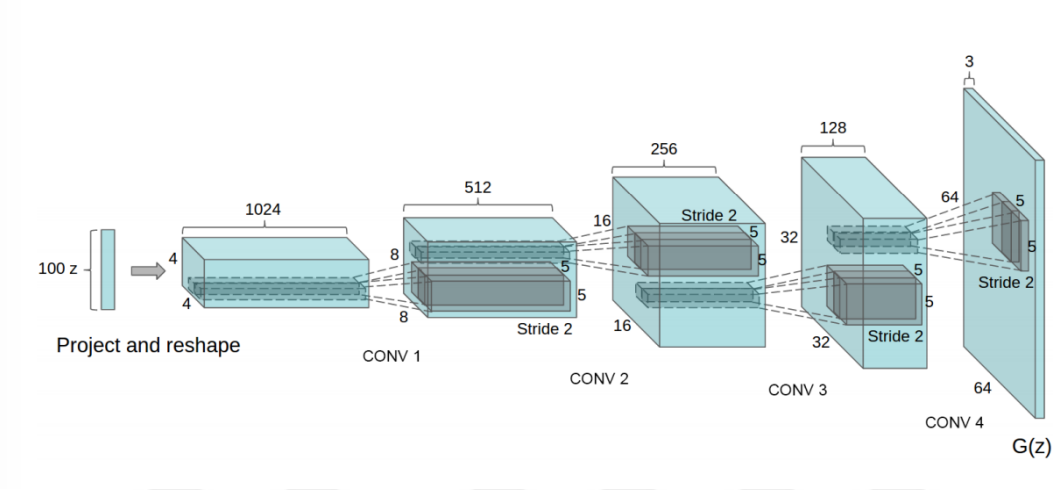
Şekil 2.15'de pix2pix ve CycleGAN algoritmalarının kullandığı veri tabanı görsellerinin örneklerini ve kullanım farkını gösteren şema bulunmaktadır.

2.6.3 DCGAN

DCGAN terimi derin evrişimli çekişmeli üretken ağlar kelimelerinin ingilizce kısaltmasından oluşmaktadır. 2015 yılında Alec Radford ve Luke Metz tarafından geliştirilmiştir (Radford ve ark., 2015). DCGAN, ÇÜA'ları eğitmek için daha kararlı mimari oluşturmaya olanak sağlamıştır. DCGAN mimarisinde pix2pix de olduğu gibi üretici ve ayrıştırıcı modeller bulunmaktadır. Üretici model bir resim uydurmaya çalışırken ayrıştırıcı ise bu resim ile gerçek resim arasındaki sahte ya da gerçek ayrımını yapmaya çalışacaktır. DCGAN mimarisinde, aktivasyon fonksiyonu olarak relu kullanılmaktadır, üretici modelin çıkış katmanı için tanh (hiperbolik tanjant fonksiyonu) kullanılır. Ayrıştırıcı model katmanlarda ise giriş negatif değerler aldığında, çıkışta doğrudan sıfır vermek yerine çok küçük negatif değerlere de izin verebilen leaky relu fonkisuonu kullanılır. Bu fonksiyonlar ile daha hızlı ve dengeli bir öğrenme aşaması hedeflenir.

pix2pix algoritması ile en önemli farklı kullanılan veri tabanıdır. DCGAN eşleşmiş

bir veri tabanı ihtiyacı bulunmamaktadır. Bu çalışma gerekli veri tabanı için eşleşmiş görüntüler kullanılmasını hedeflediğinden dolayı pix2pix algoritması ile devam edilmiştir.



Şekil 2.16 DCGAN Yöntem Diagramı (Radford ve ark., 2015).

3. BİLGİSAYAR DESTEKLİ TASARIM

Bu bölümde tasarımda yapay zekanın kullanımının alt yapısını oluşturan çeşitli tasarım yöntemleri hakkında bilgi verilmiştir. Bu tasarım yöntemlerinden hesaplamalı tasarım, algoritmik tasarım, üretken tasarım ve parametrik tasarım incelenmiş ve detaylıca anlatılmıştır. Bölümün sonunda günümüz inşaat süreçlerinde yapay zekanın rolünün nasıl olabileceği Yapı Bilgi Modellemesi başlığında aktarılmıştır.

Bu tasarım yöntemleri çalışmanın alt yapısını oluşturarak, tasarım süreçlerinde yapay zeka algoritmalarının kullanımı için ilham verici olmuştur.

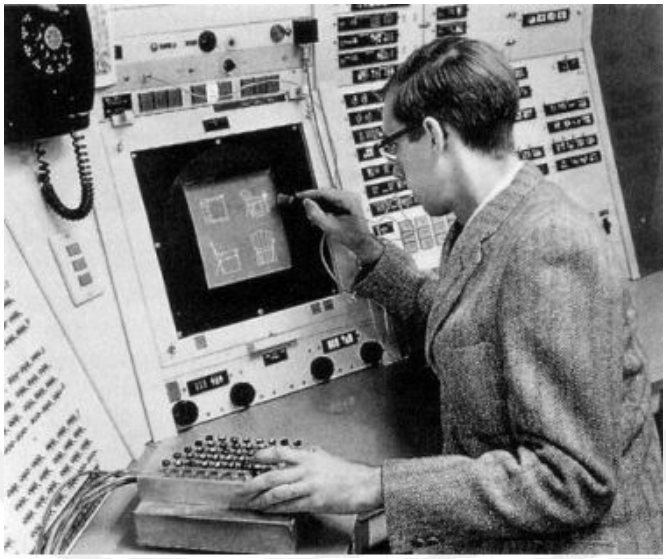
3.1 Bilgisayar Destekli Tasarım Fikrinin Doğuşu

Teknolojinin gelişmesi ile tasarım teknikleri, tasarlama fikirleri, tasarım süreçlerine dair algoritmalar zamanın koşullarına göre değişmeye başlamıştır. Bilgisayar destekli tasarım (BDT) sistemlerinin doğuşuyla tasarımların kriterleri, özelliklerinin ölçülmesi ve hesaplanması daha da yaygınlaşmıştır.

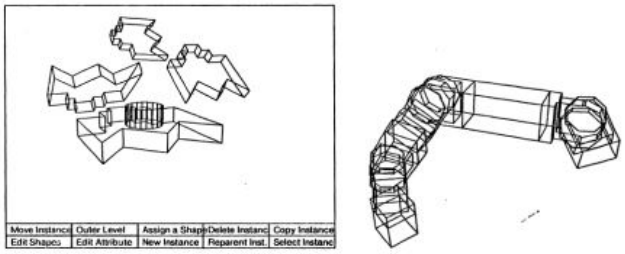
BDT sistemlerinin doğuşundan bahsederken sanayi devrimine kadar gidilebilir. Bu dönemde verim odaklı seri üretim için teknolojik araçların kullanımı fikri ile makinelere entegre çalışan otomasyon sistemleri geliştirilmiştir. 1900'lü yılların ikinci yarısında geliştirilen bilgisayar teknolojisine dayalı otomasyon sistemleri ise daha karmaşık tasarımların daha kolay üretilmesine ve kaynakların daha verimli kullanılmasına olanak sağlamıştır (Kalay, 1985).

Ivan Sutherland, 1963'te tamamladığı doktora çalışmasında "insan-makine grafiksel iletişim sistemi" olarak tanımladığı SKETCHPAD programını geliştirmiştir. O dönemde mekanik parçaların ya da elektronik devre tasarımının grafiksel olarak bilgisayara aktarılması için mühendisler yazılı komutlar kullanmak zorundaydı. SKETCHPAD sistemi ise Şekil 3.1'de görülen ışıklı bir kalem donanımını ekran üzerinden çizimi mümkün kılacak şekilde mühendisin kullanımına sunarak bilgisayar

ile tasarım fikrine yeni bir boyut getirmiştir (Sutherland, 1964). BDT sistemindeki bu gelişmeler ileriki onyıllar boyunca büyük ilerleme göstermiş yüzyılın sonlarında, mimarlık ve tasarım alanında kullanılmak üzere şekil 3.2 'de arayüzü gösterilen VEGA programı doğmuştur (Kalay, 1999).



Şekil 3.1 Ivan Sutherland SKETCHPAD sistemi arayüzünde işlem yaparken (Url-03).



Şekil 3.2 VEGA Programı arayüzü (Akküç, 2019).

3.2 Hesaplamalı Tasarım

Günümüzde tasarımcının biçim veya forma ulaşırken kullandığı en önemli yardımcı araç bilgisayar ve bilgisayar ortamıdır. Bilgisayar kullanımı, tasarımın her aşamasında artık zorunluluk haline gelmektedir. Bilgisayar destekli ya da algoritma odaklı tasarım ile tasarımcı, formu veya biçimi ortaya koyarken düşünme şeklini de ortaya koymaya başlamıştır. Klasik yöntemlerin, düşüncelerin yerine daha yenilikçi, daha alternatifli biçimlerin elde edilebileceği bir düşünce şekli oluşmuştur. Bu düşünce yapısını sayısal düşünme (computational thinking) olarak isimlendirmekteyiz (Erbaş, 2015).

Sayısal düşünme, fazlalık, hasar sınırlama ve hata düzeltme yoluyla en kötü durum senaryolarına karşı sistemi koruma ve kurtarma açısından düşünmektir. Bilgisayar bilimi için temel kavramlardan yararlanarak problem çözmeyi, sistemler tasarlamayı ve insan davranışını anlamayı içerir (Wing, 2006). Sayısal düşünme, karmaşık bir sorunu irdelememize, sorunun ne olduğunu anlamamıza ve çözümler geliştirmemize olanak tanır. Daha sonra bu çözümleri bilgisayarın, insanın veya her ikisinin de anlayabileceği şekilde aktarılabilir.

Sayısal düşünmenin dört temel tekniği vardır: Ayrıştırma , Örüntü Tanıma, Soyutlama ve Algoritma

Ayrıştırma

Karmaşık gözüken bir problemi veya sistemi yönetilebilir daha küçük parçalara bölmek olarak tanımlayabiliriz. Örnek olarak: Kütüphanede bulunan ders kitaplarını ders türlerine göre sıralama problemi. Bunun için problemi daha küçük parçalara ayırıp, iş adımlarını tanımlamamız gerekir.

1- Kitaplar kütüphaneden alınır,
2- Kitaplar ders türlerine göre gruplandırılır,
3- Kitaplar grup grup alınıp kütüphaneye yerleştirilir.

Örüntü Tanıma

Örüntü tanıma, bir makine öğrenimi algoritması kullanarak desenleri tanıma sürecidir. Örüntü tanıma, halihazırda kazanılmış bilgilere veya desenlerden çıkarılan istatistiksel bilgilere veya bunların temsiline dayalı olarak verilerin sınıflandırılması olarak tanımlanabilir. Örüntü tanımanın önemli yönlerinden biri, uygulama potansiyelidir (Ansari, 2022).

Örnekler: Konuşma tanıma, konuşmacı tanımlama, multimedya belge tanıma (MDR), otomatik tıbbi teşhis.

Soyutlama

İhtiyaç duyulanlara konsantre olmak için ihtiyaç duyulmayan fikirleri ve belirli ayrıntıları ayırma ve filtreleme süreci (Url-07).

Algoritma

Algoritma bir problemin çözümü için birbirini takip eden işlemler bütünüdür. Bilgisayar biliminin temel kavramıdır ve günümüz teknolojisi ile bütün disiplinlere etki etmektedir. Algoritmik düşünce sayesinde, yani amaca giden her bir işlemdeki performansı, artıları ve eksileri düşünme ile tasarımda bütünü oluşturan her bir girdinin değişken özelliklerine göre tasarımın yeniden analiz edilmesi ve üretilmesi mümkündür. Amaç tasarımı oluşturan girdiler arasındaki ilişkinin kurgusunu yapabilmektir (Erbaş, 2015).

(Nourian ve ark., 2013) makalesi hesaplamalı tasarımı daha iyi anlayabilmek için başarılı bir örnektir. Mimari mahal dizilimlerinin konfigüratif tasarımı için parametrik CAD programı olarak geliştirilen tasarım metodolojisi ve araç takımı tanıtmaktadır. Tasarımcıların, işlevsel alanların birbirine bağlanması için gereken yolu çizerek yerleşim planı sürecini başlatabileceği bir araç seti oluşturmuştur.

3.2.1 Üretken Tasarım

Üretken tasarım, tasarımcının tasarım hedeflerini, istenilen performans ve gereksinimlerini karşılamak için belirlenen kısıtları tasarım yazılımına verip, yazılımın tasarım alternatifleri oluşturmasıdır. Yazılım bu kısıtları ve performansı sağlayacak tüm ihtimalleri oluşturur.

(Merrell ve ark., 2010) çalışması üretken tasarım için başarılı bir örnektedir. Oyunlar ve grafik uygulamaları için mimari plan diziliminin otomatik olarak oluşturulması için bir yöntem sunulmaktadır. Makine öğrenimi ve mekan dizilimi algoritmalarından faydalanarak bilgisayara mimari yönden eksiksiz yapılar üretecek bir yöntem tanıtılmıştır.

Günümüzde mimarlık alanında tasarım sürecinde kullanılan en popüler üretken

tasarım yazılım araçları, Rhino Grasshoper[1] ve Autodesk Dynamo[2]'dur. En sık kullanıldığı amaç ise, cephe tasarımlarının belirli kısıtlar ve hedefler doğrultusunda farklı alternatiflerini elde etmektir.

3.2.2 Algoritmik Tasarım

Algoritmik Tasarım, klasik yöntemler ve klasik tasarım araçları için zorluklar yaratabilecek karmaşık geometrilerin modellenmesine imkan sağlar. Algoritmik Tasarım, tasarımın parametreler aracılığıyla manipüle edilebileceği anlamına gelen parametrik bir modelleme felsefesi gerektirir. Bu, tasarımcının daha geniş tasarım olasılık yelpazesini hızla ve daha az efor sarfederek keşfetmesini sağlar (Url-08).

3.2.3 Parametrik Tasarım

Parametrik tasarımın tanımı için literatürde geçen kabul görmüş tanımlamalar kronolojik sıraya göre şu şekildedir:

(Moretti, 1971) çalışmasında parametrik mimari olarak adlandırmış ve "boyutların birbiri arasındaki ilişki" olarak tanımlamıştır. (Kalay, 1989) parametrik modellemeyi "otomatik olarak güncellenmesi ve ekranda görselleştirilmesi" sözleri ile anlatmıştır. (Szalapaj, 2013) parametrik tasarımı "geometrik kısıtlamaların yanı sıra boyutsal ilişkiler ve verilerin kullanımı" olarak tanımlamıştır. (Kolarevic, 2003) parametrik tasarımı bir süreç olarak tanımlamış ve "bir tasarımın şeklinin değil, belirli bir özelliğinin tasarlanması" olarak ifade etmiştir. (Woodbury, 2010) parametrik tasarım için "karmaşık dijital tasarım modellerinin oluşturulması, yönetimi ve organizasyonu" ifadelerini kullanmıştır. Son yıllarda mimarların parametrik tasarımı yaklaşımlarını giderek daha fazla benimsediğini ve dolayısıyla tasarımda hesaplama yöntemleri kullanmanın avantajlarını kabul ettiğini göstermektedir (Caetano ve ark., 2020). Parametrik tasarımı, istenilen kısıtları ve gereksinimleri karşılayabilecek şekilde boyutları arasında ilişkilerinin bulunması ve bu ilişkilerin de yönetimi olarak tanımlayabiliriz. Fernando Romero tarafından tasarlanan Soumaya parametrik tasarım için güzel bir örnektir (Şekil 3.3).

Museo Soumaya, 14. yy'dan günümüze uzanan uluslararası tablolar, heykeller ve

[1] https://www.rhino3d.com/
[2] https://www.autodesk.com/products/dynamo-studio/overview

dekoratif objelerden oluşan bir koleksiyona ev sahipliği yapabilen heykelsi bir bina. Binanın dış cephesine bakıldığında her açıdan farklı bir görünüşe sahip amorf bir yapıdadır. Binanın ayırt edici cephesi, korunmasını ve dayanıklılığını kolaylaştıran altıgen alüminyum modüllerden yapılmıştır. Kabuk, her biri kendi geometrisine ve şekline sahip farklı çaplarda çelik kolonlarla inşa edilmiş ve ziyaretçi için doğrusal olmayan dolaşım yolları yaratılmıştır. Binanın tasarım sürecinde en önemli ve inşa etmesi zor bölümü cephesidir. Cephenin inşasını zorlaştıran konuları şunlardır: 1- Mekansal modeli önemli ölçüde değiştirmeden yapının gerçek formuna uyum sağlamak. 2- Kolon yüzeylerinden en fazla birkaç milimetre uzakta olması gereken müzenin hem dış hem de iç duvarlarını şekillendirmek. 3- Yüzey düzensizliklerini düzeltip ve zeminden çatıya ve binanın tüm çevresi boyunca sürekliliği sağlamak. 4- Binlerce farklı altıgenin konumunu ve yönünü kesin olarak tanımlamak. Binanın geri kalanının inşaatıyla aynı anda hızlı, kısa bir programa göre tasarlamak, üretmek ve inşa etmek. Geometrica firması geliştirdikleri program sayesinde bu sorunları çözmeyi başardılar. Lazer teknolojisini kullanarak inşaa edilen gerçek yapının geometrisini programlarına işlemiş ve cephede bulunan altıgen panelleri gerçek duruma göre modelleyerek tasarım ve inşa sürecini yönetmişlerdir (Url-05).

Şekil 3.3 Soumaya Müzesi, Fernando Romero (Url-05).

3.2.4 Mekan Dizimi

Mekan Dizimi, binaların ve şehirlerin mekansal dokularını incelemek için kullanılan teknikler bütünü ve mekan ile toplumu birleştiren bir teoriler zinciri olarak mimarlık ve kentsel tasarım alanlarındaki en etkili bilimsel çalışmalardan biridir (Hillier & Hanson, 1997).

Mekan Dizimi yaklaşımın en önemli özelliği, insan zihninde tasarladığı mekanın yansıması bilgisinin oluşmasında kritik rolü olan mekanın soyut karakteristiklerini somut olarak ifade ve analiz etmeyi sağlayabilen sayısal bir teknikdir (Kubat ve ark., 2007). Bu tekniklerde ilk hedeflenen olgu, mekansal organizasyonun bireyin eylemleri ve görüş alanları ile temas ve bağlantısını nesnel olarak etüt ederek, "mekanların" bireyleri bir araya toplama ve oryantasyon becerilerini açığa çıkarmaktır. Bahsi geçen teknik, çağımızda; mimarlık, iç mimarlık, kentsel tasarım ve planlama gibi mimari iş kollarının çalışmalarında kullanıldığı gibi öte yandan da ulaşım, arkeoloji, antropoloji, enformasyon teknolojileri, şehir ve insan coğrafyası, peyzaj mimarlığı ve bilişimle ilgili bir çok dalda da kullanılmaktadır (Gündoğdu, 2014).

Bu çalışma içinde mekan dizimi tekniğinden etkilenilerek, bir mahaldeki mekanların diziliminin matematiksel olarak nasıl hesaplanacağı üzerine araştırmalar yapılmıştır.

3.3 Yapı Bilgi Modellemesi

Yapı Bilgi Modellemesi (YBM), yapının tasarımının, inşasının, mühendislik analizlerinin ve işletilmesinin gerektirdiği tüm bilgileri dijital olarak temsil etmesi için Akıllı Dijital Model (ADM) üretme sistemidir.

Görselleştirme, çevresel şartlar altında deneyimle; enerji analizi, çakışma analizi, kod standartları kontrolü, maliyet tahmini, inşa edilen ürün bilgisi, bütçeleme ve diğer birçok amaç için bir tesisin dijital modelini oluşturma eylemidir (NIBS, 2007).

YBM, yapı ile ilgili grafik (geometri/biçim vb.) ve alfasayısal (malzeme,maliyet, fiziksel çevre kontrolü vb.) veriden oluşan üç boyutlu ADM meydana getirerek, bunu paydaşların ortak veri ortamında kullanılmasını sağlar (Ofluoğlu, 2014). YBM sisteminin bu özelliğinden kaynaklı olarak tüm performans gereksinimlerin ve ihtiyaçların belirtildiği bir projenin plan ve kütle tasarımlarında algoritmalar daha fa-

zla rol almaya başlamıştır. İlerleyen süreçte algoritmalar geliştikçe YBM sistemi ile daha uyumlu çalışacağını ve projelerin tasarım aşamasından yapım aşamasına kadar olan süreçle ilgili tüm çizim ve tasarımları bilgisayarlar programlarının ve algoritmaların üstleneceği öngörülmektedir.

3.3.1 Akıllı Dijital Modeller

YBM sisteminin çıktılarından en önemli olanı ADM'dir. Bu modelleri yapının dijital bir kopyasının üretildiği modellerdir. Yapıya dair her elemanın dijital olarak fonksiyonel ve fiziksel özellikleri ile üretilmiş halidir. Örnek olarak: Yapıya ait bir duvar elemanının fonksiyonel ve fiziksel özellikleri bulunmaktadır. Fonksiyonel özellikleri iki mahal arasında ayırıcı bir katman görevini görmesi, içinde bulunan elemanlarının ses ve ısı yalıtımı sağlaması veya güvenlik amaçlı bir bölme olabilir. Fiziksel özellikleri ise yüksekliği, uzunluğu, kalınlığı, rengi içerisinde bulunan boşluklar olabilir. Yapıya dair tüm elemanların bu şekilde modellenmesi ADM'leri oluşturmaktadır.

3.3.2 Dijital İkiz

Günümüzde nesne ve sistemlerin üretim, inşa veya kurulum işinden önce genellikle bilgisayar ortamlarında deneyimlenmektedir. Bilgisayar ortamında deneyimlenen bu nesnelerin performans analizi, maliyet analizi, üretilebilirlik, inşa edilebilirlik gibi verileri nesne fiziksel olarak veya sistem devreye girmeden dijital ortamlarda test edilmektedir. İşte bu nesnelerin bilgisayar ortamında oluşturulan dijital kopyalarına dijital ikiz denilmektedir.

Dijital ikizlerden sıklıkla şu şekilde faydalanılmaktadır: Nesnelerin ve sistemlerin olası hatalarını dijital ikizlerini deneyimleyerek keşfetmek maliyeti azaltır, kaliteyi arttırır. Koordinasyonu arttırarak daha iyi bir iş birliği ile daha doğru işler yapımasını sağlar. Projenin başlangıç ve bitiş zamanlarını daha doğru ön görebilmekte böylelikle zamanı daha iyi kullanmaya yardımcı olmaktadır.

Tezin amacının bağlamında dijital ikizler bu çalışma için önemlidir. Yapay zeka algoritmaları ile tasarlanan projelerin inşa edilebilmesi için YBM sistemine adapte olması şarttır. Fiziksel olarak inşa edilecek yapıların projelerini YBM ortamında yapay zeka algoritmaları ile tasarlamak sistemin bütüncül ve tutarlı çalışmasını sağlayacaktır.

4. ŞEMATİK PLAN TASARIMINDA ÇEKİŞMELİ ÜRETKEN AĞLAR

Bu bölümde mimari şematik plan tasarımında yapay zeka algoritmalarının nasıl kullanılabileceği ve sürece ne ölçüde yardımcı olabileceği konusu üzerine çalışmalar yapılmıştır.

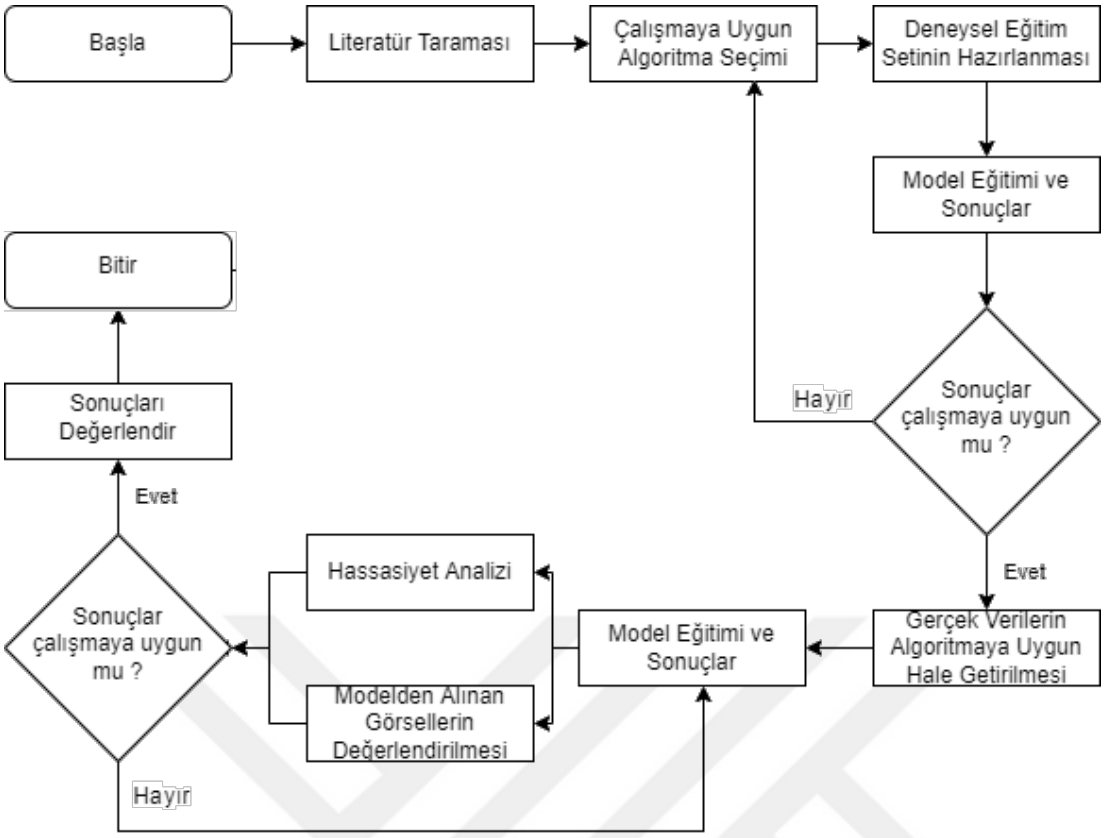
Konutların yeniden kullanımına ve kentsel dönüşüme yönelik mimari şematik plan üretiminde kullanılmak üzere apartman daireleri gibi küçük ölçekli projeler üzerinden araştırma yapılmıştır. Bu ölçekteki ve detaydaki planlar, daha karmaşık planlar için altyapı oluşturmak ve araştırmadan daha hızlı sonuçlar almak için seçilmiştir.

Bu çalışma kişisel ihtiyaçlar doğrultusunda, standartlara ve bölgesel yapı mimarisine uygun şematik planların seçilen veri tabanına bağlı kalarak üretilmesini mümkün kılacak algoritmik tasarım örneklerini de ele almaktadır.

Çalışmada kullanılan planlar yapıdaki bağımsız bölüm sayısına ve oda sayısına göre kategorize edilmiştir, böylelikle daha genel ve daha spesifik bir model eğitimi ile bunların karşılaştırılması mümkün olmuştur.

4.1 Yöntem

Çalışmanın nasıl bir yol izlediği bu bölümde anlatılmıştır. İlk olarak model eğitiminde kullanılan verisetlerinin algoritmaya uygun bir şekilde nasıl düzenlendiği ve toplandığı anlatılmıştır. Daha sonra modelin nasıl eğitildiği ve pix2pix algoritmasının nasıl şematik plan üretimi yaptığının anlatımıyla devam edilmiştir. Son olarak üretilen şematik tasarım planları incelenmiştir. Bu süreci anlatan iş akış şeması aşağıdaki gibidir (Şekil 4.1).



Şekil 4.1 Çalışmanın Yöntemine Yönelik İş Akış Şeması

Çalışma aşamasında pandemi şartlarından ötürü firmalar ile iletişim kısıtlı şartlar altında yapılmış ve veri tabanına ulaşımı olumsuz bir şekilde etkilemiştir. Ayrıca model eğitiminde kullanılan bilgisayarın özelliklerinden dolayı da (i7 8565U işlemci ve 12GB ram) model eğitiminde zamanı daha verimli kullanabilmek için veri tabanının niceliklerinde ve model eğitiminde kısıtlamalara gidilmiştir. Bu kısıtlar; pix2pix algorimasının devir sayısı, veri tabanında kullanılan plan sayısı ve bu planların boyutlarıdır.

4.1.1 Veri Tabanının Toplanması ve Düzenlenmesi

Makine öğrenimi için en önemli araç veri tabanıdır. Veri tabanının yeterli büyüklükte olması ve tutarlı bilgi içermesi modelin eğitiminde daha istikrarlı kararlar verebilmesi için önemlidir. Bu çalışma için gerekli olan eğitim veri tabanı konut dairelerinin mimari kat planlarıdır. Veri tabanında bulunan mimari planlar İstanbul Ataşehir bölgesine ait ilgili kurumlarca onaylı mimari ruhsat projeleridir. Ataşehir bölgesinde kentsel dönüşüm ve yeniden kullanıma yönelik pix2pix algoritmaları ile mimari şe-

matik plan kullanımı bu çalışmanın özgün yanıdır, literatürde ilk kez bu çalışma ile Ataşehir bölgesi için pix2pix algoritmaları kullanılarak mimari şematik plan oluşturmak için araştırma yapılmıştır. Araştırmanın sonucunu daha iyi analiz edebilmek için eğitim verilerinde derlemeler ve kategorizasyonlar yapılmıştır.

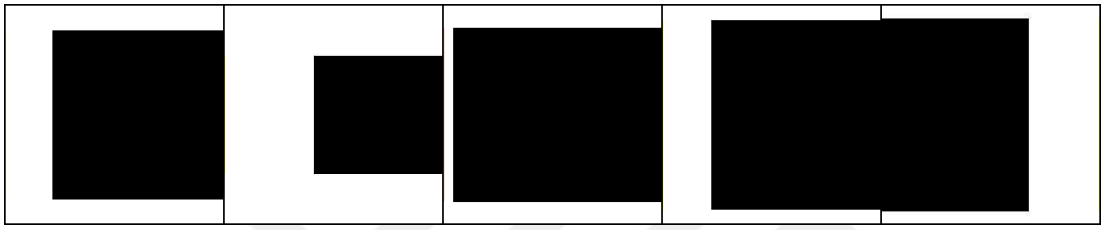
Mevcut eğitim verilerinde bulunan mimari planlar, daire tiplerine ve katta bulunan daire sayılarına göre kategorilere ayırılmıştır. Bu kategoriler şu şekildedir:

- Katta tek daire,
- Katta iki daire, dairelerin oda sayısı farklı,
- Katta iki daire, daireler 2+1 tip,
- Katta iki daire, daireler 3+1 tip,
- Katta üç daire, dairelerin oda sayısı farklı,
- Katta üç daire, daireler 1+1 tip,
- Katta dört daire, dairelerin oda sayısı farklı,
- Katta dört daire, daireler 1+1 tip,
- Katta dört daire, daireler 2+1 tip,
- Katta dört daire, daireler 3+1 tip,
- Katta beş daire, daireler 2+1 tip,
- Katta beş daire, daireler 3+1 tip,
- Katta beş daire, dairelerin oda sayısı farklı,

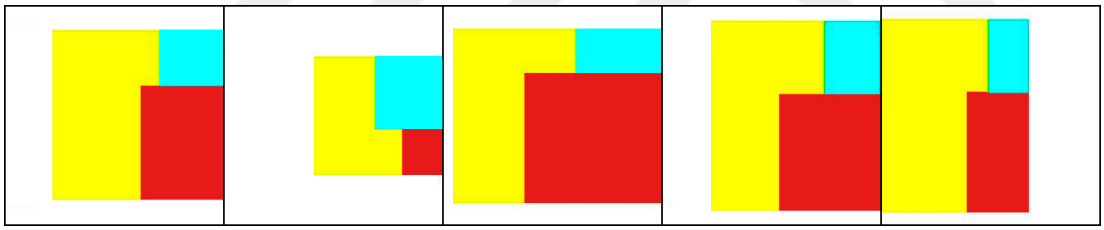
Bu kategorilere göre dört farklı model eğitimi sağlanmıştır. Bu modellerinden birincisi, veri tabanınında bulunan tüm mimari şematik planlar ile d480 devir sayısına kadar eğitilmiştir. İkinci model aynı eğitim verileri ile devir sayısında değişiklik yapılarak d700 devir sayısına kadar eğitilmiştir. Üçüncü model katta sadece iki daire bulunan planlar kullanılarak d480 devir sayısı ile eğitilmiştir. Diğer model ise yine katta iki daire bulunan planlar ile d700 devir sayısına kadar eğitilmiştir. Bu sayede

model eğitimi sırasında kullanılması gereken verilerin içerikleri ve devir sayılarının model eğitimine olan etkisi incelenmiştir. Katta iki daire bulunan planlar, bu çalışma için anlaşılması kolay yeterli seviyede detaylı olan planlar olduğundan dolayı eğitim için seçilmiştir. Kullanılan şematik planlar Ek.A'da bulunmaktadır.

Bu çalışmada gerçek bir veri tabanı ile çalışmaya başlamadan önce algoritmayı hızlı bir şekilde test edebilmek için yapay olarak, basitleştirilmiş mimari şematik plan eğitim veri tabanı oluşturulmuştur. Şekil 4.2'de modele tanıtılacak ilk girdi görselleri, şekil 4.3'de ise ilk girdi görselleri ile eşleştirilmiş şematik plan görselleri bulunmaktadır.



Şekil 4.2 Deneysel eğitim verileri ilk girdi örnekleri.



Şekil 4.3 Deneysel eğitim veri tabanında bulunan şematik plan örnekleri.

Bu planlar python programlama dilinde hazırlanan bir algoritma ile oluşturulmuştur. Her renk ait olduğu fonksiyonu temsil etmektedir. Oluşturulan bu veri tabanında planları birbirinden ayıran en önemli özellik kapladığı piksel alanının geometrisidir. Her bir planın kapladığı piksel alanının geometrisi benzersizdir. Bunun nedeni model eğitimi sırasında aynı geometriye sahip ilk girdi görsellerinin iki farklı hedef görseline gitmesini önlemektir. Bu sayede daha tutarlı model eğitimi hedeflenmiştir. Yeterli miktarda üretilen planların ardından pix2pix modeli eğitilmeye başlanmıştır. Eğitimi tamamlanan pix2pix modeline verilecek olan ilk girdi görselleri; eğitim veri tabanı içerisinde bulunmayan piksel alanının geometrisi veri tabanında bulunanlardan benzersiz görsellerdir. Bu modelden alınan sonuc neticesinde çalışmaya gerçek veri tabanı

kullanılarak devam edilecektir. Deneysel planların üretimi ile İlgili algoritmaya ait kaynak kodlar EK.B içerisinde yer almaktadır.

Görsel dosyaların oluştrulması ve kaydedilmesi ile ilgili gerekli kütüphanelerin yüklenmesini sağlayan kaynak kod Ek A.1'de gösterilmiştir.

Ek A.2'da deneysel planların oluşturulma algoritması gösterilmiştir. "model" olarak tanımlanan fonksiyon iki adet görsel oluşturmaktadır. İki görselin de boyutları aynı pixel değerlerine sahip olacaktır, birinci görsel (image1) sadece siyah, diğer görsel (image2) ise kırmızı, sarı ve mavi renklerden oluşacaktır. Bu görsellerin boyutları modele verilen sayılara bağlı olarak değişecektir. Görsellerin boyutlarına ait parametreler "a, b, width, height, c" dir. Oluşturulan görseller ayrı iki klasöre kaydedilir.

Ek A.3'da planların benzersiz olmasını sağlayan kontrol ve kaydetme aşamasına ait kodlar gösterilmiştir. Bu planların boyutları ve hedef görselin kırmızı, mavi, sarı renkleri belirli bir aralıkta rastgele verilen parametre değerleri ile oluşturulur. Aynı boyutta birden fazla örnek olmaması için kontrol (control) listesi oluşturulmuştur. Görsellerin boyutları küçültülerek daha hızlı sonuç alınması sağlanmıştır.

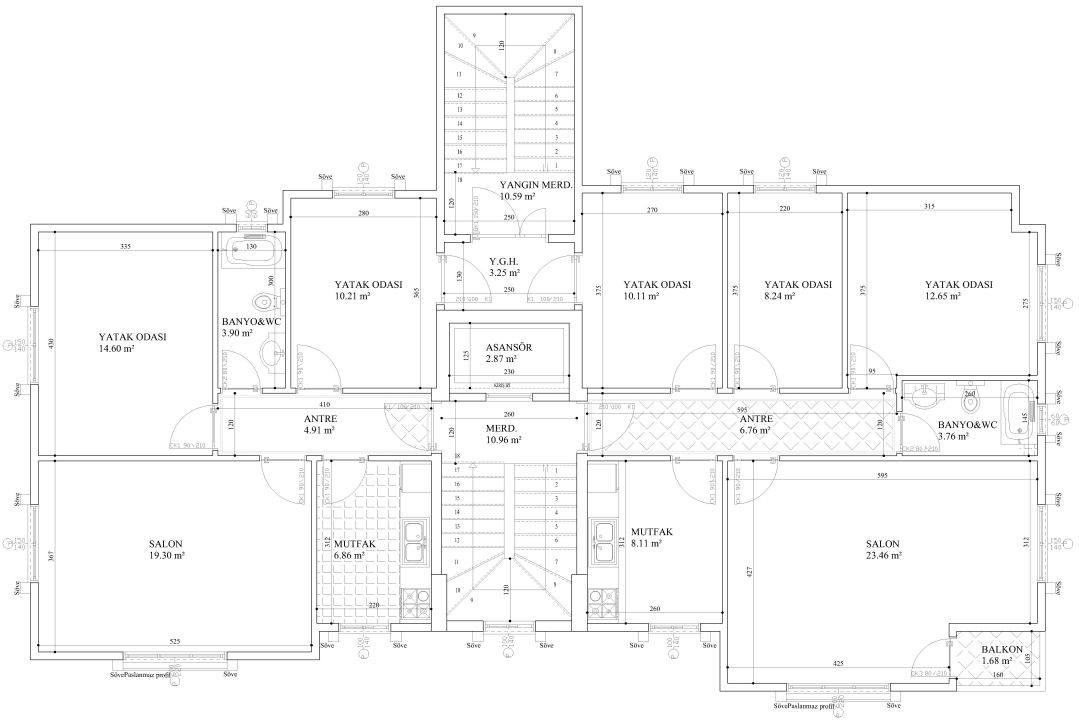
Bu sayede model için gerekli veri tabanının benzeri hızlı bir şekilde hazırlanmıştır. Ayrıca modelin öğrenmesinin kontrolünü daha hızlı test etme fırsatı bulunmuştur.

Bu aşamadan sonra kullanılacak planlar daha önce uygulaması yapılmış veya proje onayı almış planlardan oluşturulmuştur (lila Mimarlık, 2021). Alınan örneklerde anonimlik korunmuştur. Bu planların daha önce belediyeler ve ilgili yetkililer tarafından onaylı Ataşehir bölgesi için çizilmiş mimari planlar olmasına önem verilmiştir. Ayrıca kat planlarını bağımsız bölüm sayısına göre gruplara ayrılmıştır. Örnek olarak mimari kat planında iki daire var ise iki daireli kat planları, tek daire var ise tek daireli kat planları olarak gruplanmıştır. Bu sayede modelin şunları öğrenmesi sağlanmıştır;

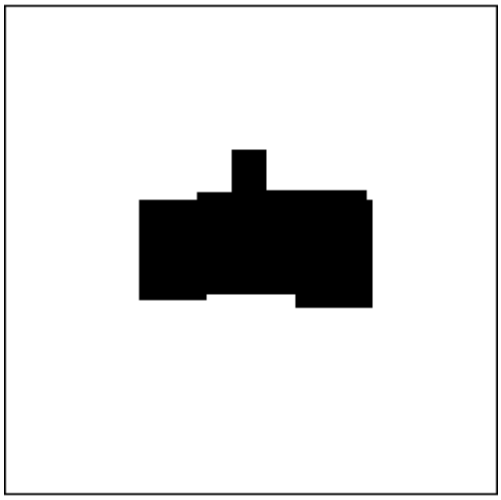
- Belirlenen muhite özel standartlar, kurallar varsa bunları kategorize yöntemi ile öğrenecektir.
- Aynı çevrede bulunan binalar ile benzer tipolojiye sahip mimari kat planları üretilecektir.
- Mimari kat planında istenilen bağımsız bölüm sayısına göre uzmanlaşmış mod-

eller oluşturulacaktır.

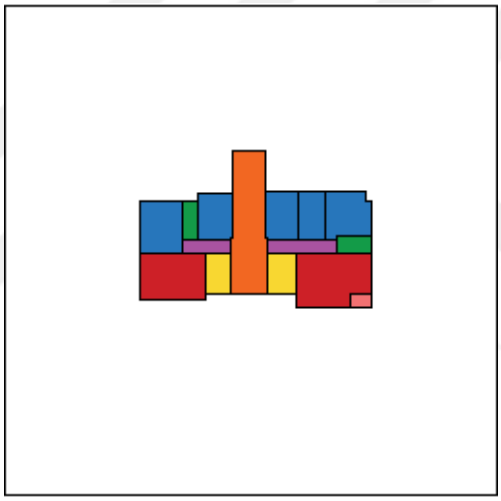
Veri tabanında mimari tüm çözümleri yapılmış ve ilgili kurumlar tarafından da onaylanmış 150 adet mimari kat planı bulunmaktadır. Bu veri tabanını pix2pix algoritmasına uygun bir biçimde tekrar düzenlemek gerekiyordu. Şekil 4.4'de bulunan mimari kat planından üretken modele girdi olarak verilecek ilk girdi verisi türetildi (Şekil 4.5). Mimari kat planının dış sınırlarına sadık kalarak içi siyah renk ile boyandı. Bunun nedeni hedef görüntü ile girdi görüntüyü mimari kat planı sınırları özelinde eşlemektir. Pix2pix algoritmasında ihtiyaç olan eşleştirilmiş görsellerden biri siyah renge boyanmış bu mimari kat planlarıdır. Son olarak veri tabanındaki kat planlarını odaların fonksiyonlarına göre her odaya ayrı bir renk vererek renklendirildi (Şekil 4.6). Bu sayede hangi odaların nasıl dizilmesi gerektiğini modele öğretmek amaçlanmıştır. Algoritmanın eşlenmiş diğer görselleri de bu planlardan oluşmaktadır. Renklendirilmiş şematik planda renklerin hangi fonksiyonları temsil ettiği şekil 4.7'de belirtilmiştir. Bu işlemler Autodesk Autocadprogramı ile yapılmıştır.



Şekil 4.4 26 No'lu Örneğin Mimari Kat Planı Verisi.



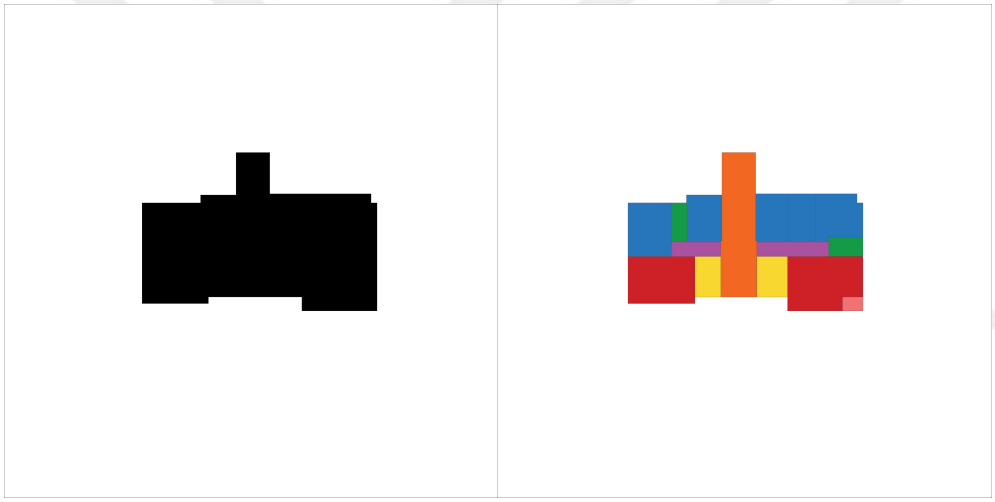
Şekil 4.5 26 No'lu Örneğin İlk Girdi Verisi.



Şekil 4.6 26 No'lu Şematik Kat Planı.

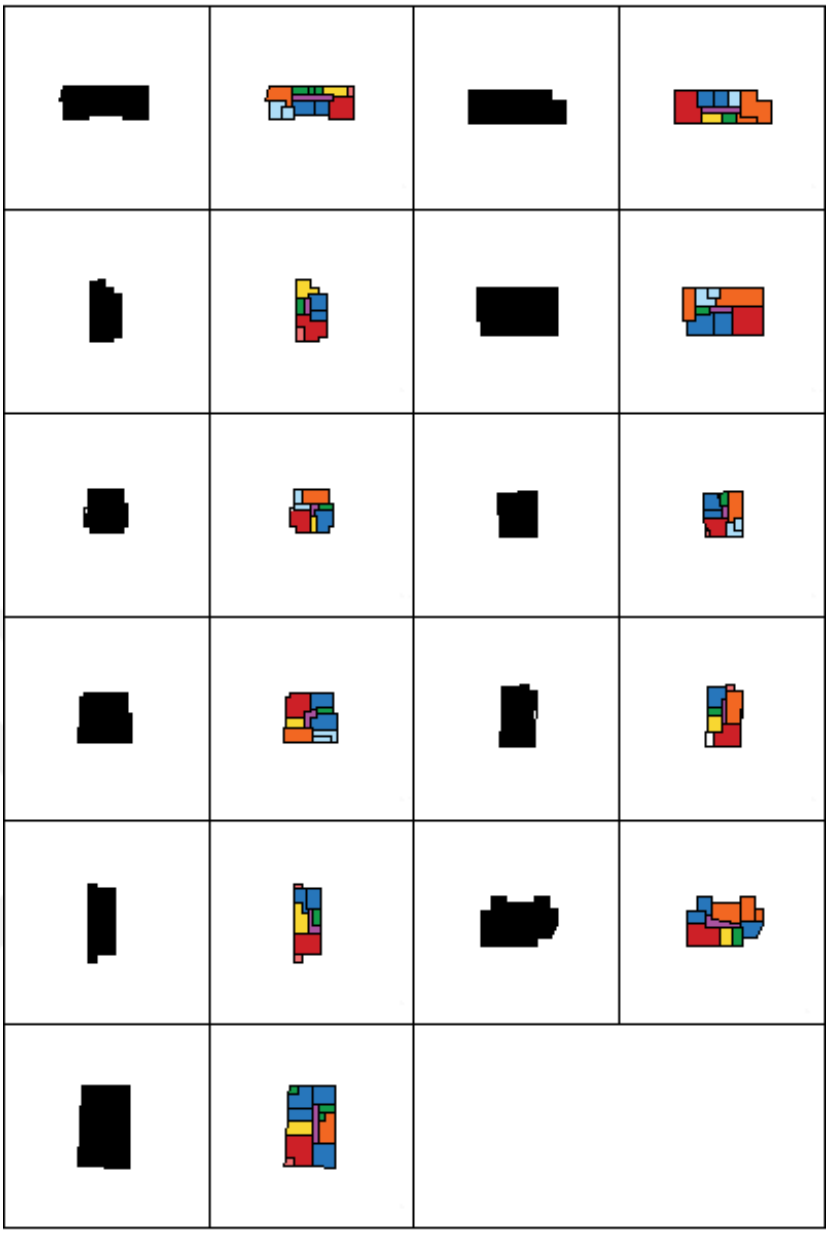
Hem ilk girdi hem de şematik planların boyutları 256x256 piksel olarak ayarlanmıştır. Bunun boyut model eğitiminde kullanılan bilgisayarın hızına bağlı olarak optimum değer olarak belirlenmiştir. Veri tabanındaki her örneğin ilk girdi ve şematik plan verisini eşleştirerek üretici ve ayrıştırıcı modellerin eğitimi sağlanmıştır. Şekil 4.8'de bulunan eşleşmiş olan bu görselin boyutu ise 512*256 olarak modele verilmiş, ilk girdi kısmının üretici modele, şematik planın ise ayrıştırıcı modele girdi olarak verilmesi sağlanmıştır.

Şekil 4.7 Renk tanımları.

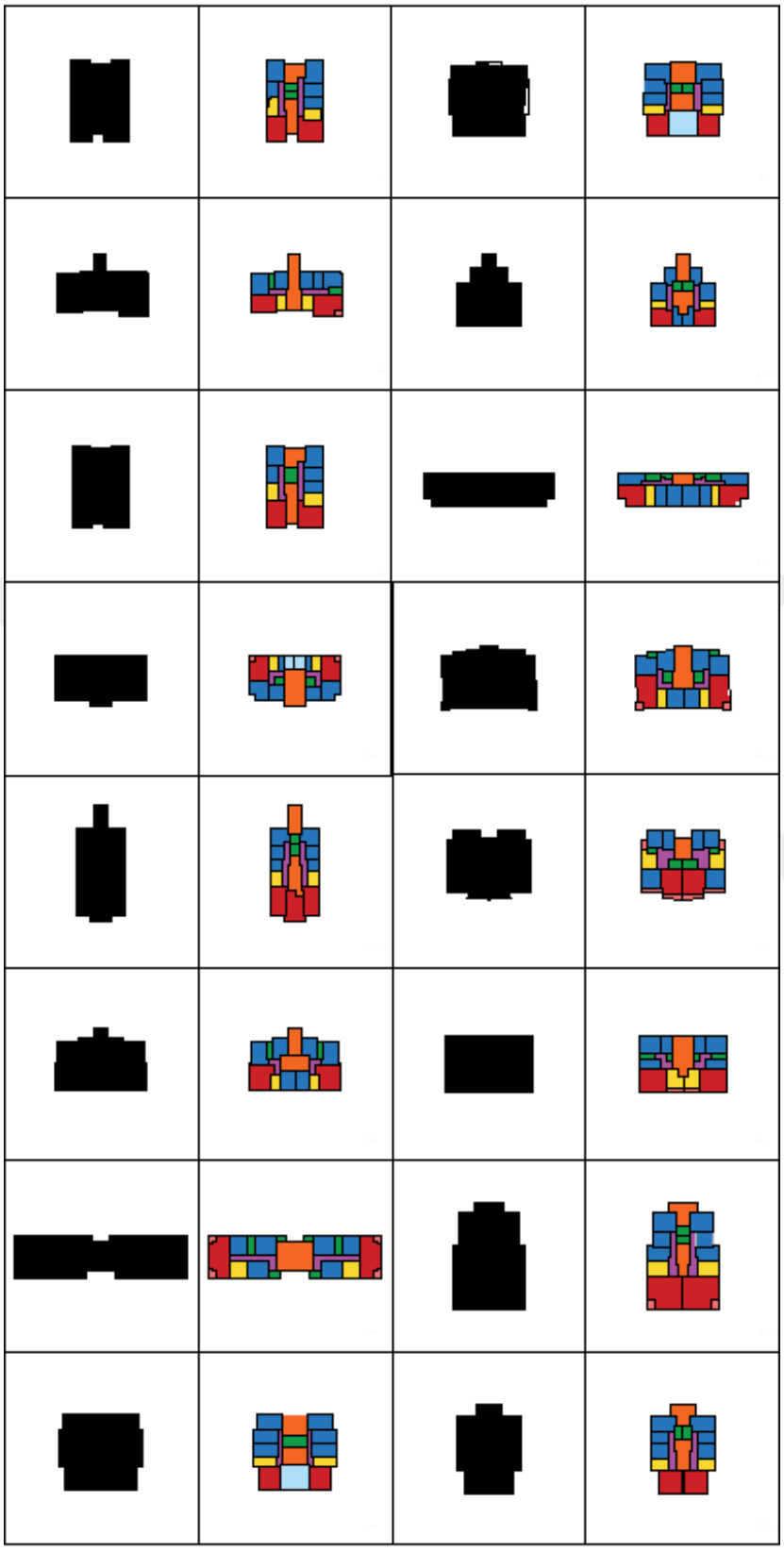


Şekil 4.8 26 no'lu örneğin ilk girdi ve şematik plan eşleştirilmiş girdi verisi.

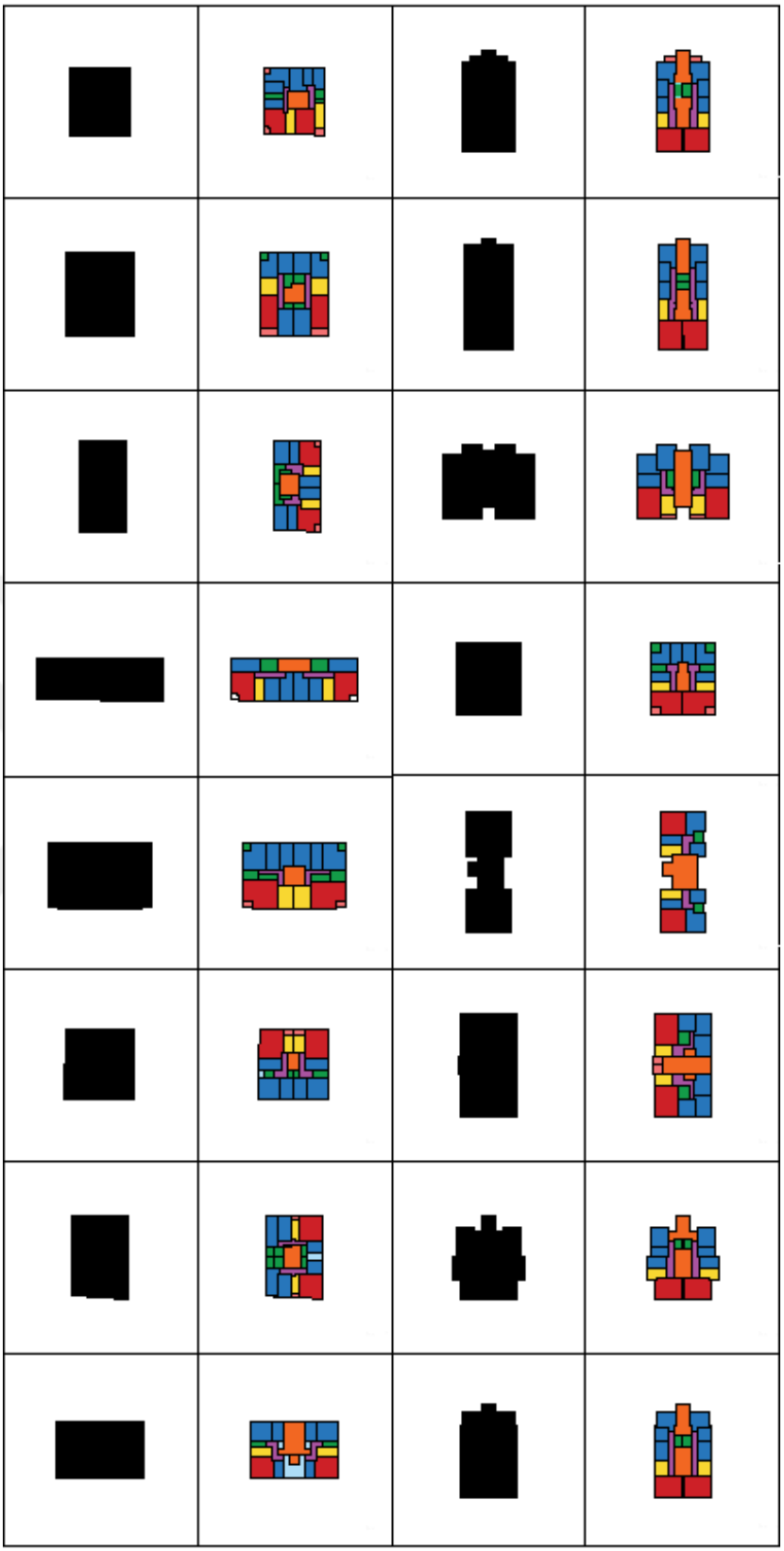
Bu çalışmada eğitim verisi olarak kullanılan projelerin şematik plan haline getirilmiş örnekler aşağıdaki gibidir (Şekil 4.9-4.20).



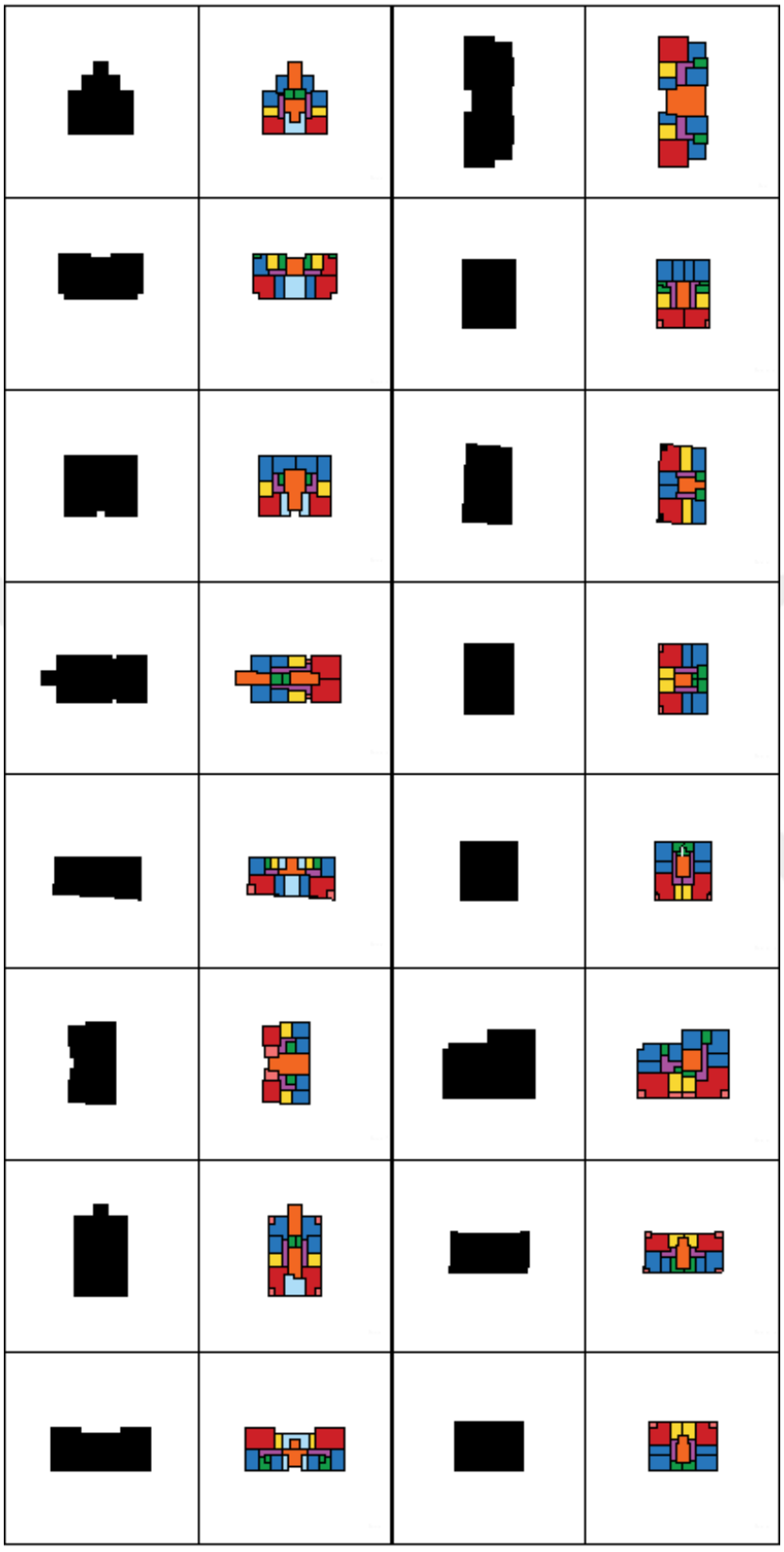
Şekil 4.9 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-Tek Daireli Kat Planları.



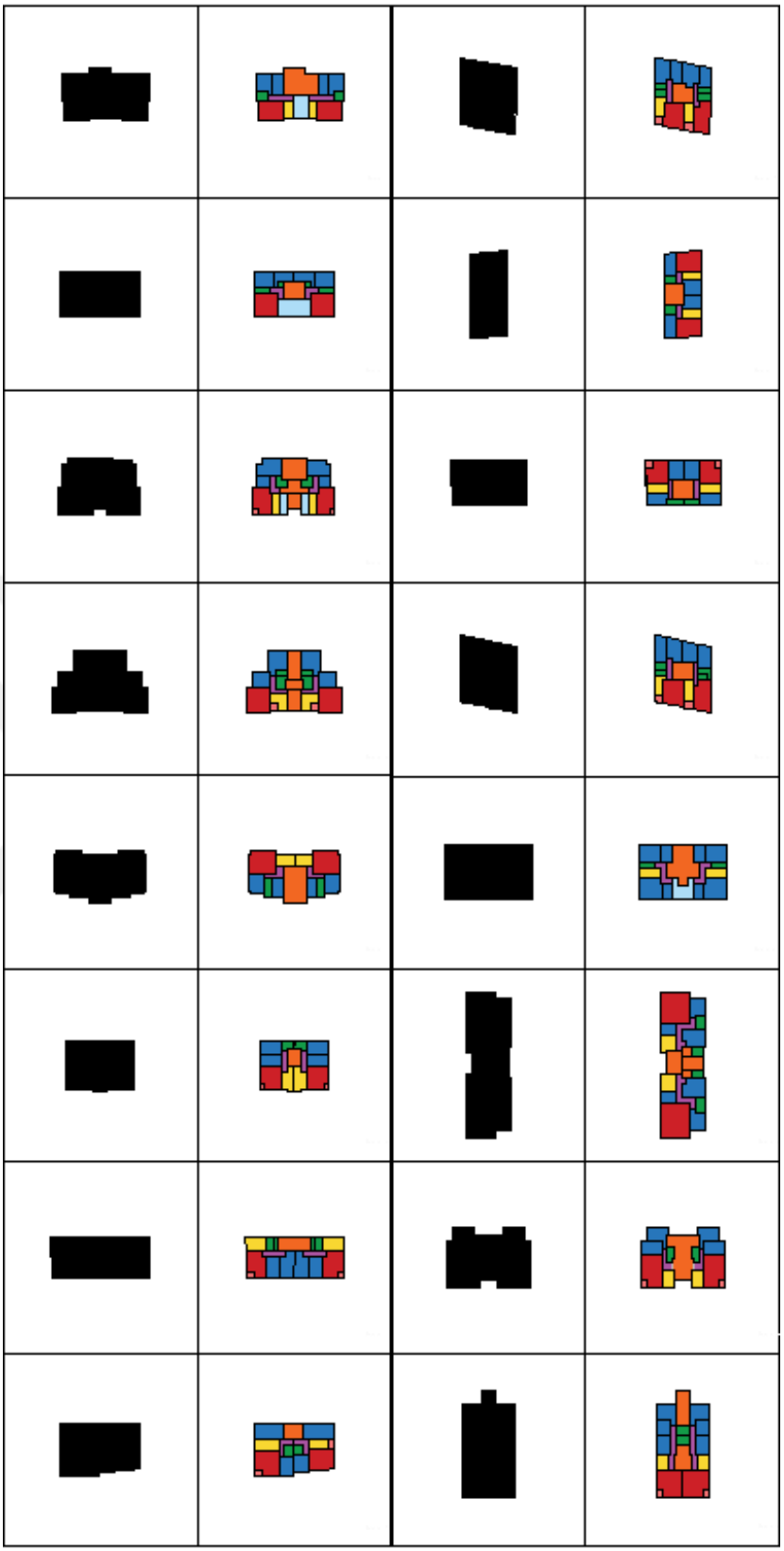
Şekil 4.10 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-İki Daireli Kat Planları-1.



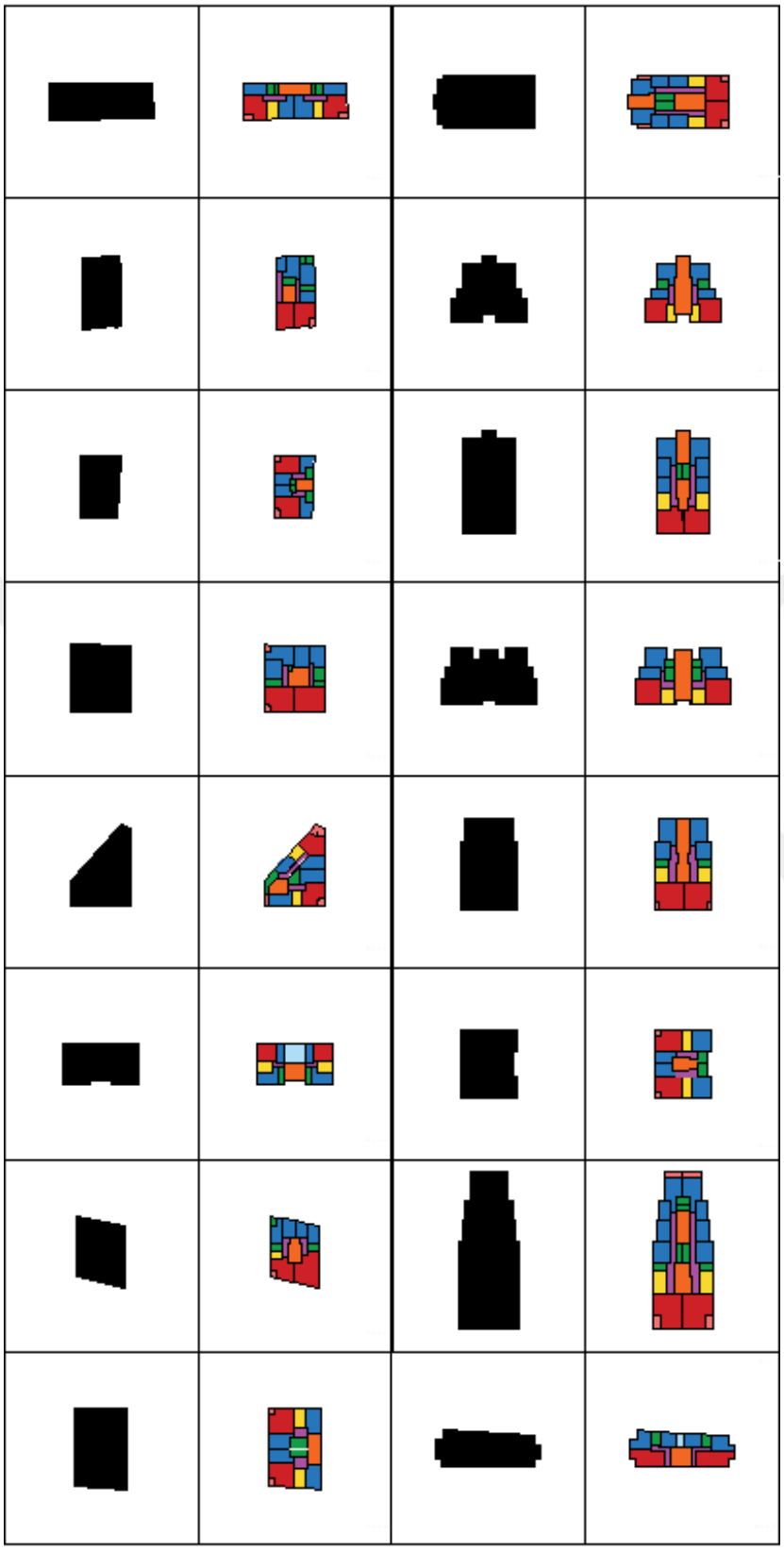
Şekil 4.11 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-İki Daireli Kat Planları-2.



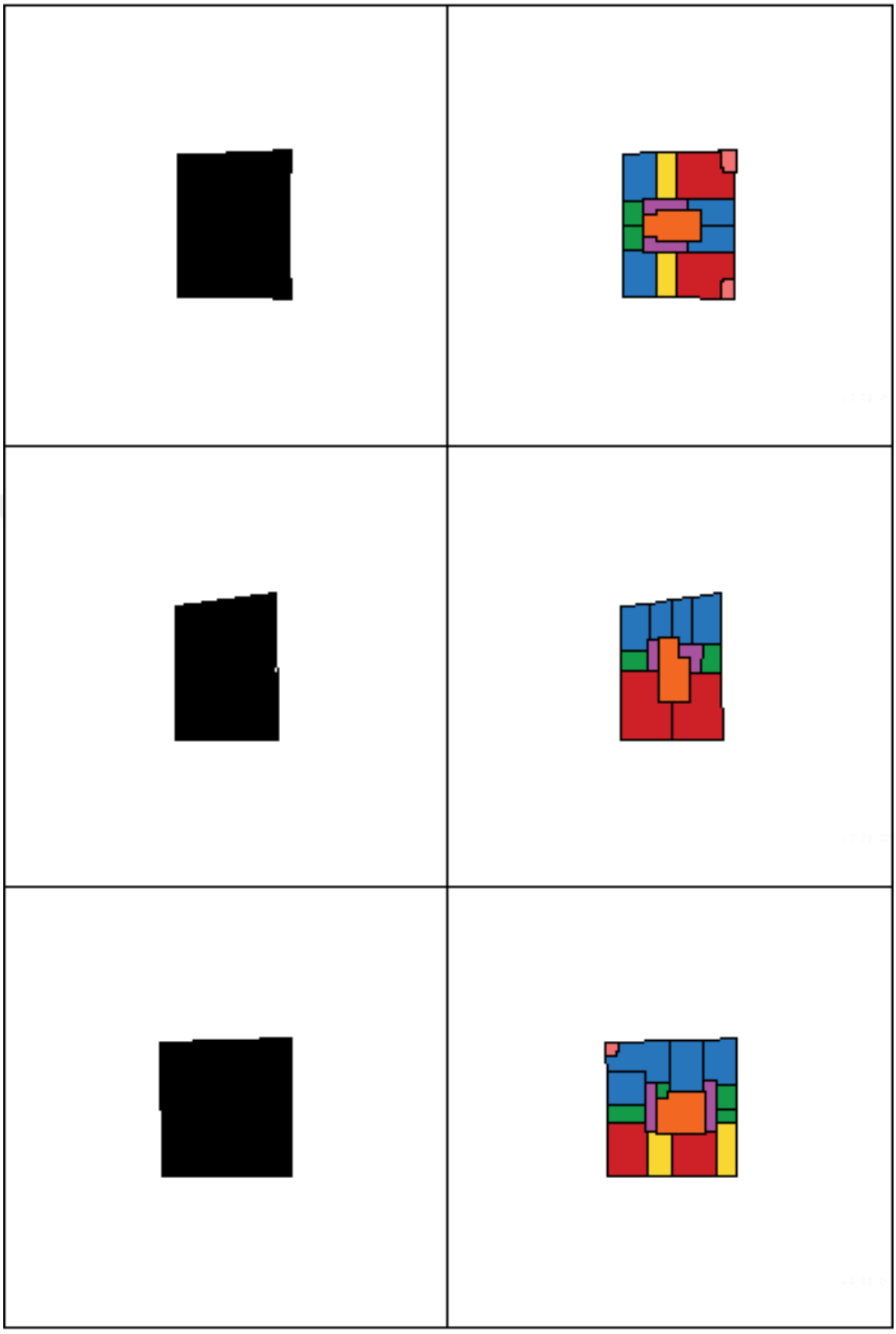
Şekil 4.12 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-İki Daireli Kat Planları-3.



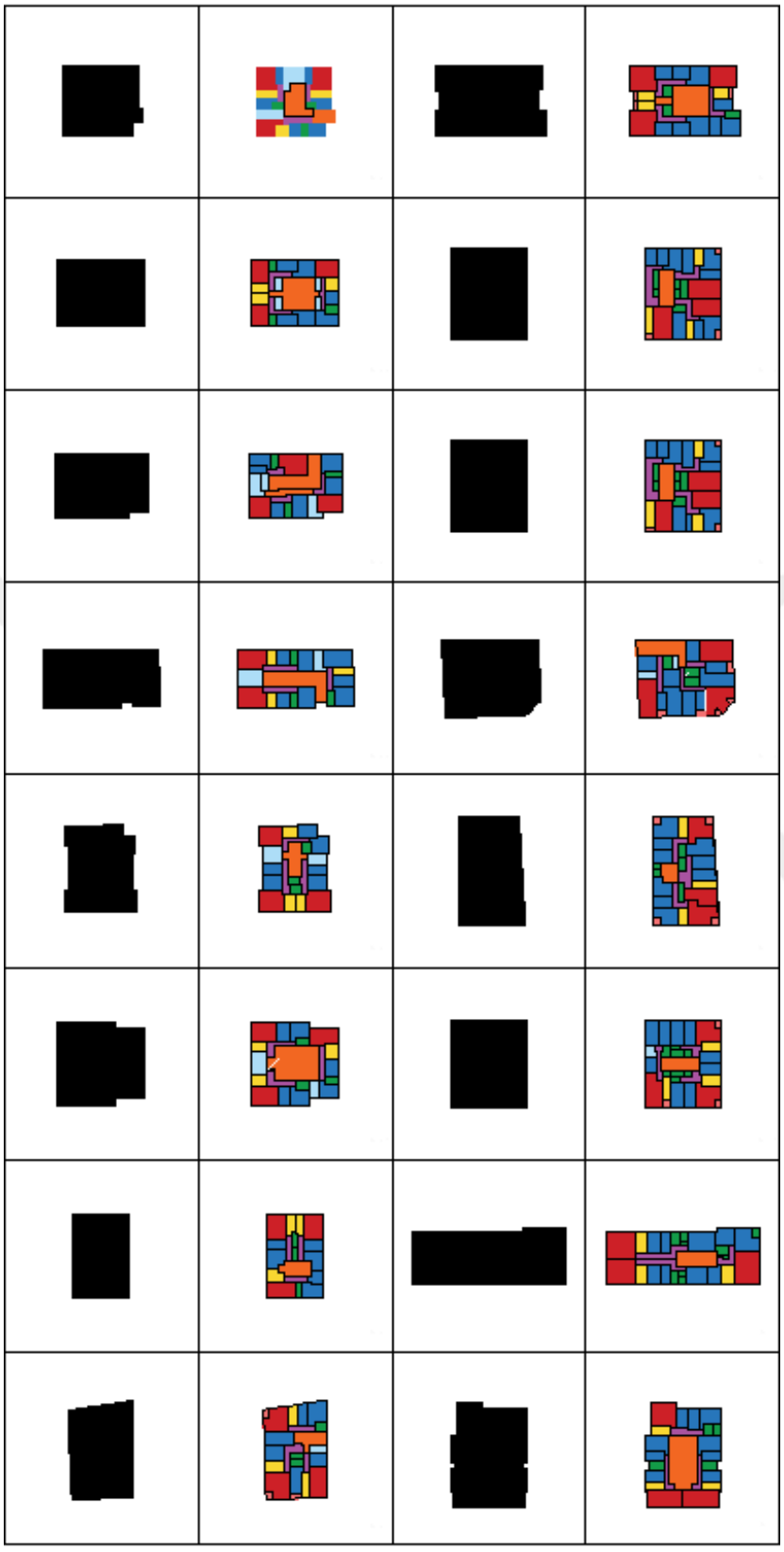
Şekil 4.13 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-İki Daireli Kat Planları-4.



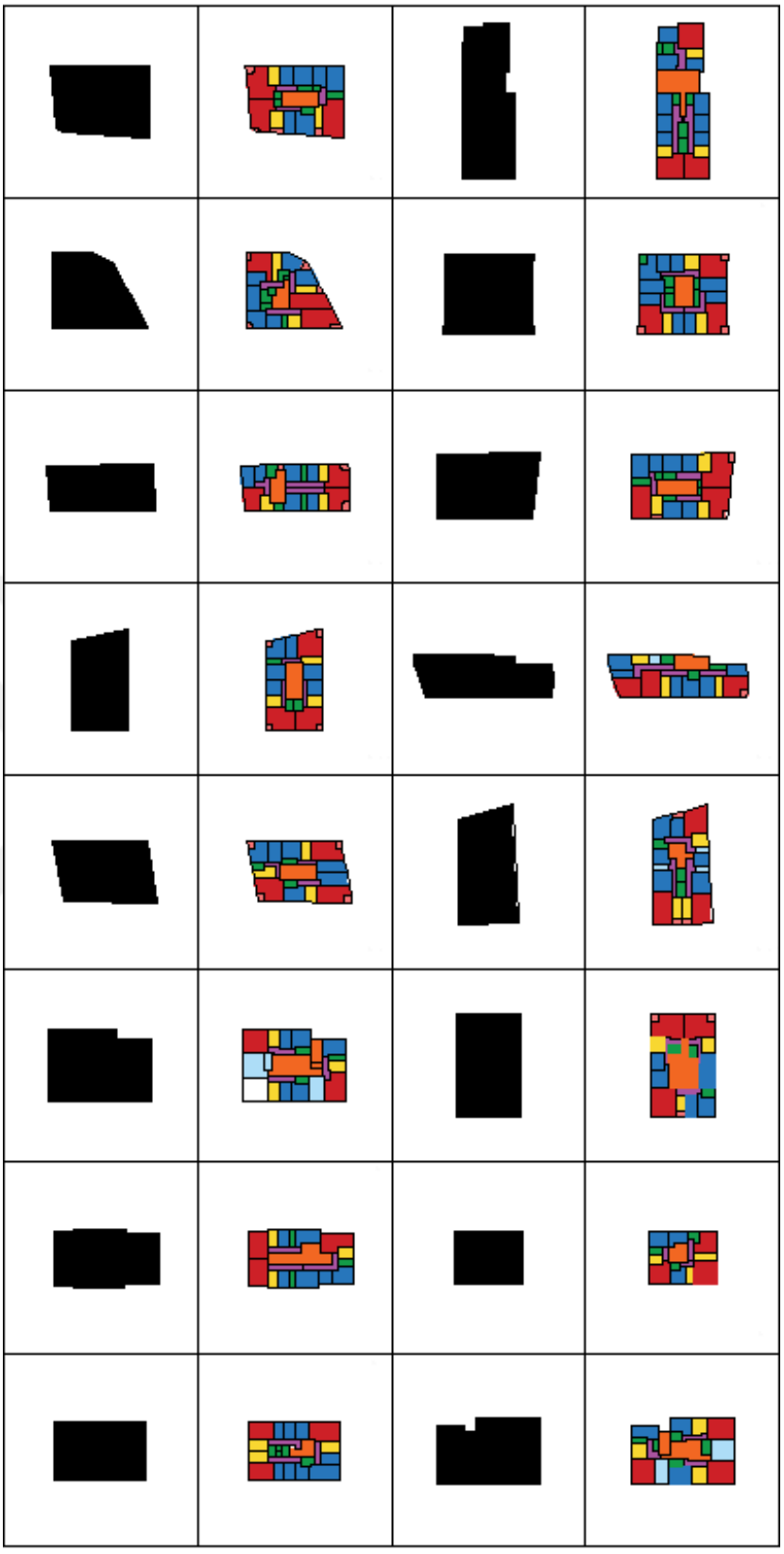
Şekil 4.14 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-İki Daireli Kat Planları-5.



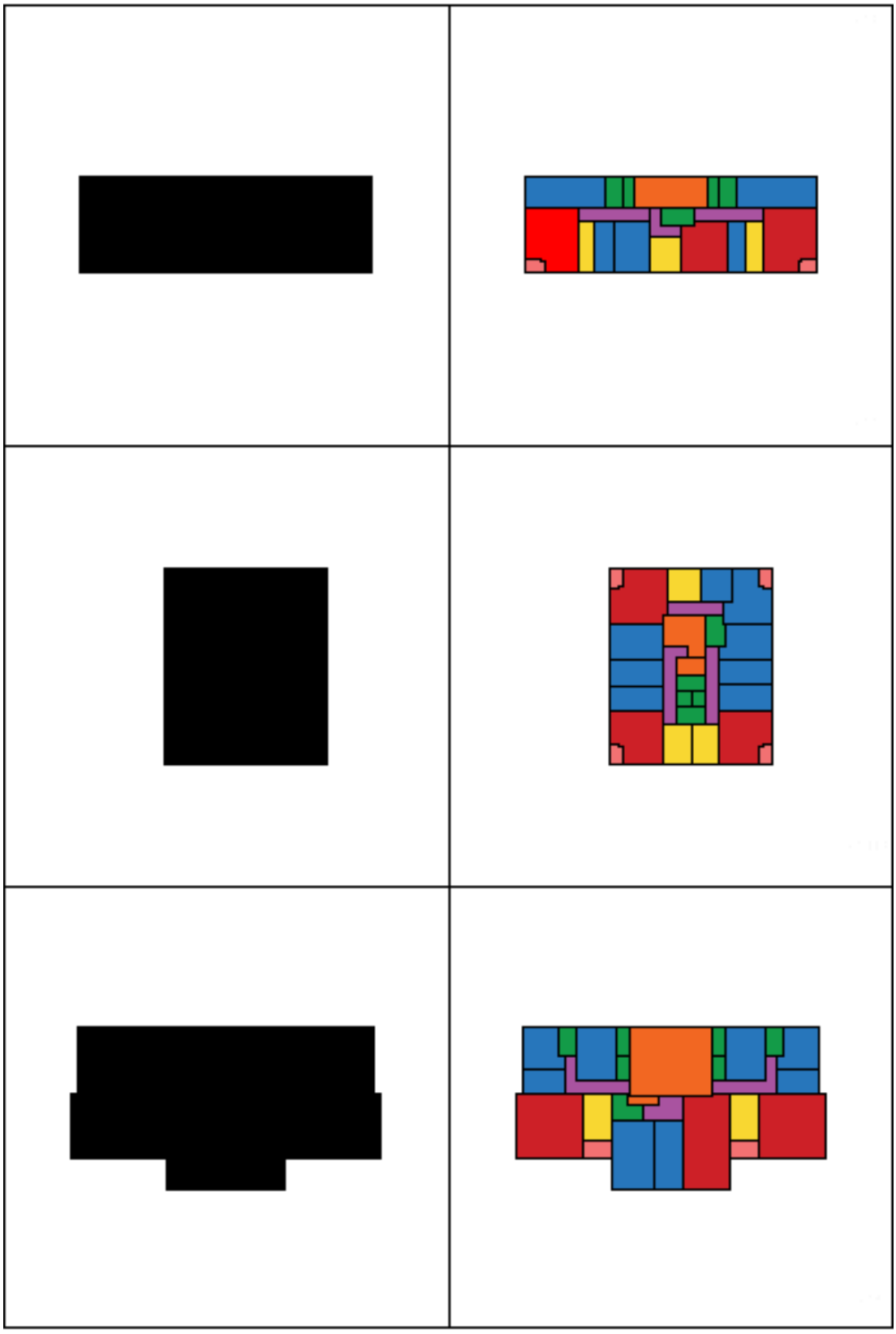
Şekil 4.15 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-İki Daireli Kat Planları-6.



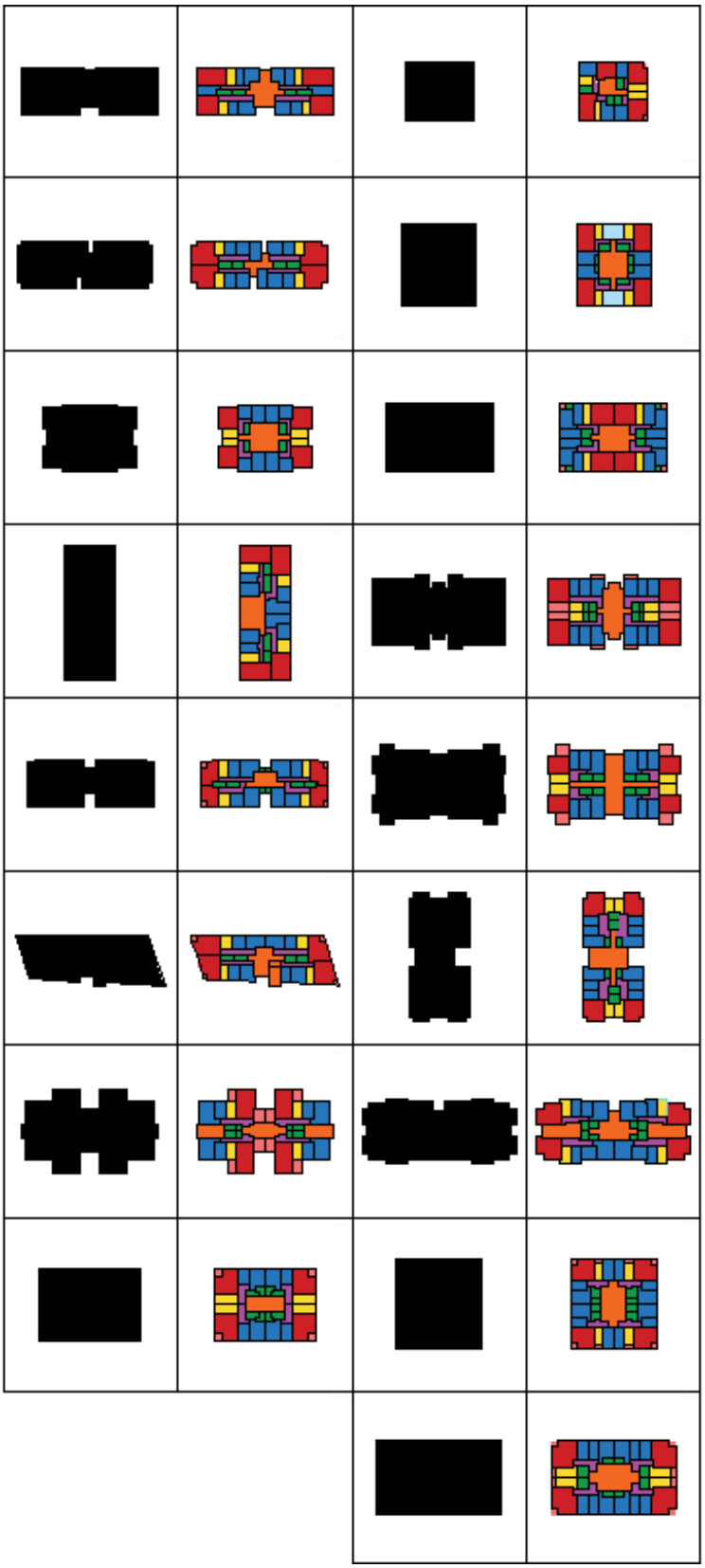
Şekil 4.16 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-Üç Daireli Kat Planları-1.



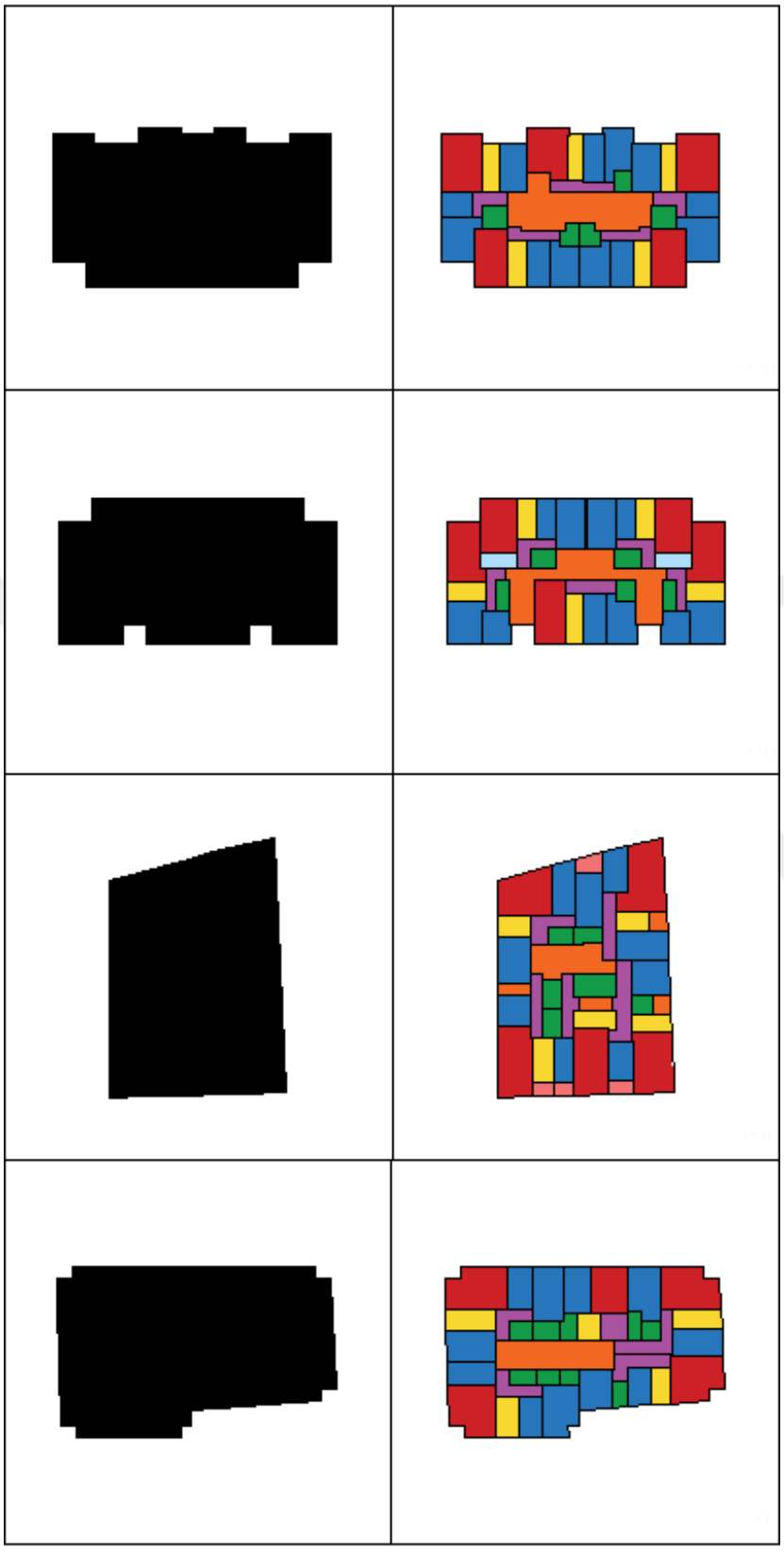
Şekil 4.17 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-Üç Daireli Kat Planları-2.



Şekil 4.18 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-Üç Daireli Kat Planları-3.



Şekil 4.19 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-Dört Daireli Kat Planları.



Şekil 4.20 Eğitim Verisinde Bulunan Eşlenmiş Şematik Planlar-Beş Daireli Kat Planları.

4.1.2 Çekişmeli Üretken Ağ Modelinin Eğitimi

ÇÜA modeli python programa dili ile PyTorch (Url-01) ve NumPy (Url-06) kütüphanesi kaynak kodlarından ve dökümanlarından faydalanarak pycharm[1] entegre geliştirme ortamında oluşturulmuştur.

Pix2pix algoritmasına ait kaynak kodları bu çalışmada veri tabanı konumunun ve modelden çıkan verinin kaydedilmesini sağlayan ilgili kaynak kodları ekte gösterildiği gibidir (Ek A.4).

Üretken modelin eğitimini gerçekleştiren kaynak kodları ekte gösterildiği gibidir (Ek A.5).

Ayrıştırıcı modelin eğitimini gerçekleştiren kaynak kodları ekte gösterildiği gibidir (Ek A.6).

Modellerin, öğrendikçe geliştirdikleri parametrelerin kaydedildiği ve model çıktılarının kaydedilmesini sağlayan kaynak kodlar ekte gösterildiği gibidir (Ek A.7).

Modelin öğrenme hızı, kullanacağı datalar ile ne kadar tekrar öğrenme yapacağı ve girdi verisi olarak kullandığı görsellerin boyutları ile ilgili parametrelerin ayarlandığı parametreler bulunmaktadır. Modelin Öğrenme aşaması süresince "NUM_EPOCHS = 1000" , "SAVE_MODEL = True" , "LOAD_MODEL = False" olarak ayarlanmaktadır. Model parametrelerini güncelleştirdikçe kaydedecektir ve mevcut veri tabanındaki görselleri kullanarak 1000 tekrar yapacaktır, bu durumda veri tabanında 150 örnek olduğu için batch size (toplu iş boyutu) 1000*150=150000 olacaktır. Öğrenme aşaması bittikten sonra ise "NUM_EPOCHS = 1", "SAVE_MODEL = False" , "LOAD_MODEL = True" olarak değiştirilerek, modelin ilk girdi verisi için öğrenilmiş modeli kullanması sağlanır. İlgili kaynak kodları ekte gösterildiği gibidir (Ek A.8).

512*256 piksel boyutundaki ilk girdi ve mimari şematik plan eşleşmiş görseli (Şekil 4.8) girdi verisi olarak algoritmaya dahil olur. Görselin 0*256 ile 256*256 piksellik kısmı üretken modele ilk girdi verisi olarak, 256*256 ile 512*256 piksellik kısmı ayrıştırıcı modele gerçek veri veya hedef veri olarak verilir.

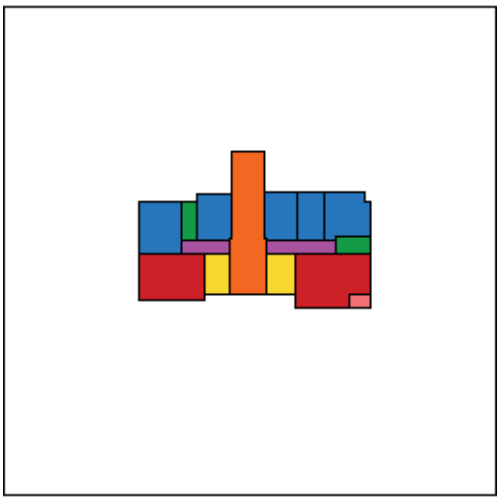
[1] https://www.jetbrains.com/pycharm/

Eğitim veri kümesinde 150 görüntü vardır. Bir epoch (devir), bu örnek sayısı üzerinden bir iterasyondur, batch size (toplu iş boyutu) 150 eğitim adımı anlamına gelmektedir. Generator her 5 devir veya her 750 eğitim adımında kaydedilir ve değerlendirilir. Model 1000 devir veya toplam 150000 eğitim adımında çalışır.

Üretken model ilk girdi verisini kullanarak hedef görseli hiç görmeden onunla ayırt edilmeyecek derecede yakın bir görsel oluşturabilmek için çalışır. İlk tahmininde daha önce gerçek görsel ile ilgili parametrelerini hiç güncelleştirmediği ve veri tabanı hakkında hiç bilgisi olmadığı için oluşturduğu görsel başarısız olacaktır. Bu oluşturulan görsel ayrıştırıcı modele girdi verisi olarak verilir.

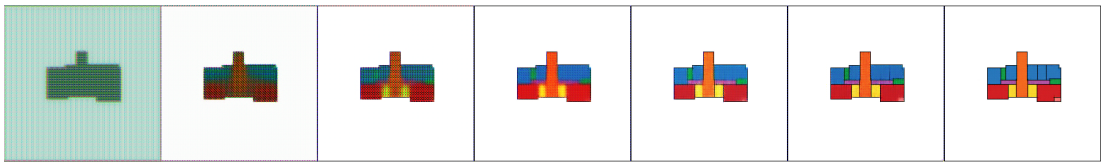
Ayrıştırıcı model, üretken modelinden gelen görselin gerçek mi yoksa sahte mi olduğunu analiz eder ve analiz sonucunda çıktı olarak üretken modele gerçeklik değeri verir. Ayrıştırıcı modelden gelen veri ile parametreleri güncelleyen üretken model daha gerçekçi modeller üretmeye başlar. Böylelikle her bir döngüde üretken model gerçeğe daha yakın görseller oluşturmaya, ayrıştırıcı model ise ayırt etmesi daha zor görseller için gerçeklik değerleri üretmeye başlar. Bu mimari sayesinde üretken model ve ayrıştırıcı model eş zamanlı olarak daha doğru kararlar vermeyi öğrenirler.

Aşağıda bir eğitim süresince üretken modelin hedef görsel (Şekil 4.21) için oluşturduğu görselleri yer almaktadır (Şekil 4.22).



Şekil 4.21 Hedef Görsel

Aşağıdaki şekil 4.22 görseli üretken modelin şekil 4.21 hedef görseli için oluşturduğu 1., 11., 76., 113., 156., 215. ve 480. devirdeki görsellerdir.



Şekil 4.22 Eğitim aşamasındaki üretken modelin çıktıları

Yukarıda anlatılan algoritmaya ilişkin projede kullanılan kaynak kodları ekte gösterildiği gibidir (Ek A.9).

Model eğitimi aktifleştiren kod ise ekte gösterildiği gibidir (Ek A.10).

4.1.3 Pix2pix Algoritması ile Şematik Plan Üretimi

Plan üretimi aşamasında modele girilmesi gereken bir ilk girdi görseli gereklidir. İlk girdi görseli istenilen alanın dış konturlerinden itibaren siyah dolgulu şekilde gözüken bir görsel olmalıdır. Bu ilk girdi görselleri aşağıdaki ihtiyaçlardan dolayı oluşturulmuş veriler olabilirler:

- Arazi sınırları içerisinde belirlenen emsal hesapları yapılmış, dış konturleri de hesaplanmış bir alana yapılacak dairenin şematik plan tasarımı için
- Mimar şematik plan tasarımı bilgisi olmayan kişilerin şematik plan ihtiyacı duyduğu alanlar için herhangi bir mimari tasarım bilgisi gereksinimi duymadan şematik plan tasarımları için
- Yeterince örnek ile eğitilmiş modellerin, belediyeler gibi ilgili kurumlarınca referans olabilecek nitelikte tasarımlar oluşturulmasında kullanılmak üzere

4.2 Pix2pix Algoritması Model Performans Değerlendirmesi

Pix2pix algoritmasın tasarım performansını değerlendirmek ve hassasiyet analizi yapabilmek için algoritmanın değişkenlerine farklı değerler verilerek karşılaştırma yapılmıştır. Bu değişkenler devir sayısı ve algoritmayı eğiten veri tabanının içeriğidir. Üretici modelden gelen görsellerin anlamlı olmaya başlaması ile ilk devir sayısı d480 olarak belirlenmiş, ikinci devir sayısı ise üretici modelin aşırı eğitime (over fitting)

maruz kalmaması adına ve algoritmadan çıkan görseller de göz önünde bulundurularak d700 olarak belirlenmiştir. Bu koşullara ek olarak modelin eğitimini yapan bilgisayarın özelliklerine göre süreyi göz önünde bulundurularak d700 devir sayısı belirlenmiştir. Eğitim veri tabanı içeriğinde yapılan karşılaştırma ise sadece iki daireli planların ve katta karışık daire sayısı bulunan şematik planların olduğu görseller ile ayrı ayrı eğitim yapılmıştır.

Bu değerlendirmeler ışığında modellerin kategorize edilerek hedefe yönelik eğitim yapılması sonucun doğruluğunu ve tutarlılığını arttırmaktadır. Örnek olarak: Modelden üretmesi beklenen plan katta 3 daire olması ise eğitime veilen veri tabanı katta 3 daireli planlar olmalıdır. Katta bulunan daire sayısına karar verilmesi modelden bekleniyorsa veri tabanı katta karışık daire sayısı bulunan planlardan seçilmelidir ve model bu planlar ile eğitilmelidir.

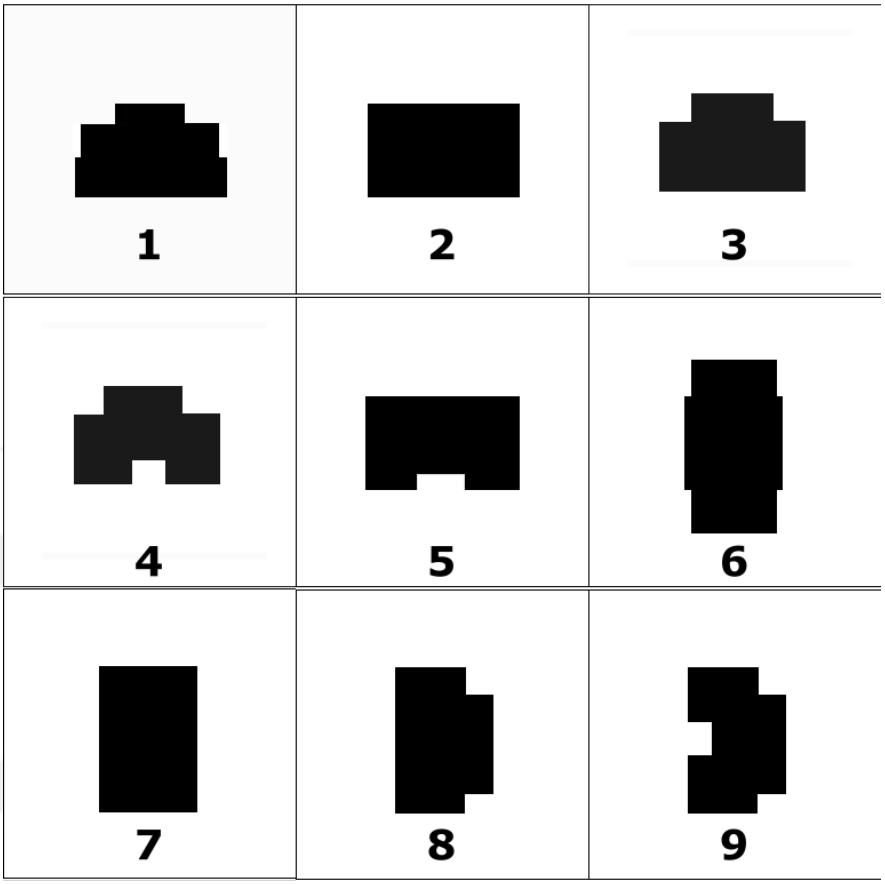
Tasarım performansını değerlendirmek ve hassasiyet analizi yapabilmek için farklı değerler verilen diğer değişken devir sayısıdır. Devir sayısının değerine karar verme aşamasında kullanılan veri tabanınında bulunan görsellerin sayısı, bu görsellerin boyutu ve kategorize edilip edilmediği devir sayısının değerinde belirleyicidir.

Bu çalışmada devir sayısının belirlenmesinde modelin anlamlı görüntüler üretmeye başladığı d480 devir sayısı ile üreticiden gelen çıktılar incelenmiş ve model eğitiminin tamamlanması için d480 devir sayısının yetersiz olduğu kanısına varılmıştır. Bundan dolayı eğitime devam edilerek d700, d1000, d1500 devir sayısına kadar eğitim devam ettirilmiştir. Bu devir sayısı ile eğitilmiş modelin ürettiği görseller birbirleri ile karşılaştırılmıştır. Mimari açıdan bakıldığında d700 ve d1000 devir sayısı üretilmiş görsellerin daha anlamlı planlar olduğu odaların ve fonksiyonların daha keskin çizgilerle ayrıldığı anlaşılmaktadır. d1500 devir sayısı ile üretilen planların ise modelin aşırı eğitiminden kaynaklı anlamsız ve eksik planlar ürettiği tespit edilmiştir.

Yukarıda belirtilen değişkenler ve bu değişkenlerin aldığı değerler ile sekiz farklı model eğitimi yapılmıştır. Modellerden alınan görsellerden dokuzar tane örnek alınıp karşılaştırma yapılmıştır. Buna göre katta iki katlı daire bulunan planlar ile d700 ve d1000 devirde eğitilmiş modelden gelen örneklerin daha okunabilir ve tutarlı planlar oluşturduğu görüşüne varılmıştır. Çalışmanın kullandığı veri tabanının optimum devir sayısı d700 ile d1000 arası civarlarında olduğu tespit edilmiştir. Çalışmanın sonuçları

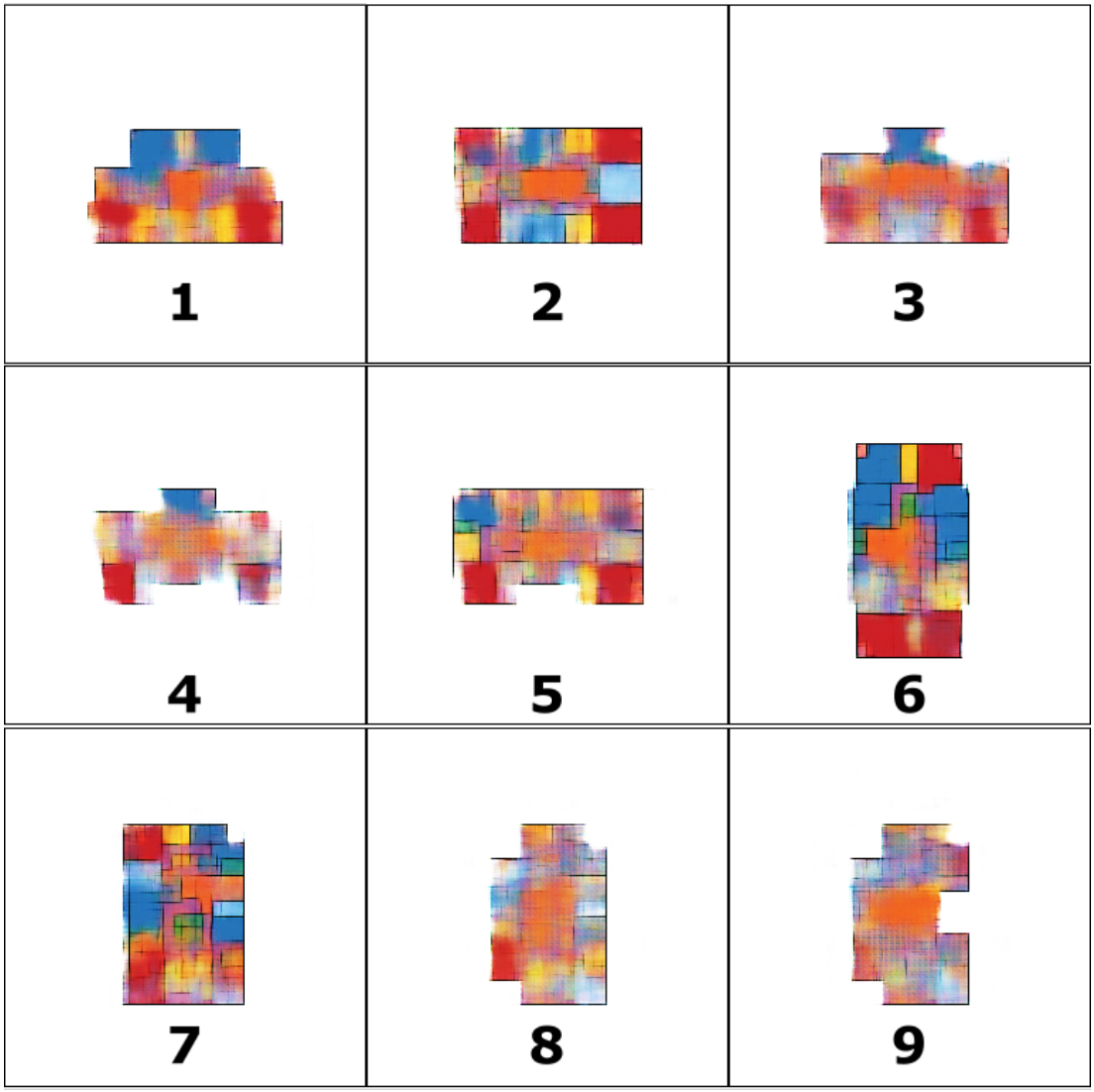
bu devir sayıları ile üretilen planlar üzerinden değerlendirilecektir.

Üretici modele girdi olarak kullanılan görseller şekil 4.23'de gösterilmiştir.



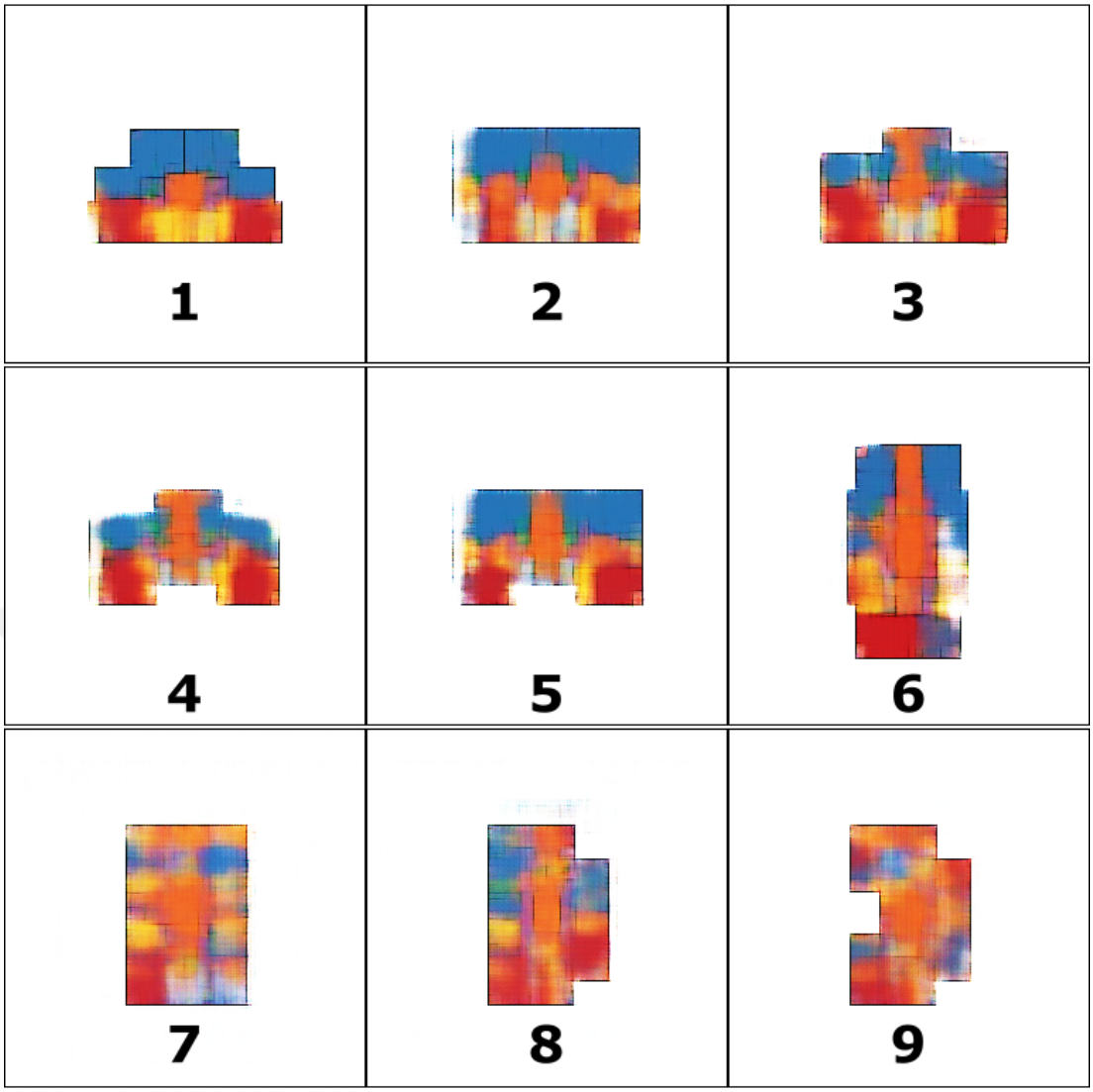
Şekil 4.23 Üretici Model Girdi Görselleri

Aşağıda d480 devir sayısı ile katta karışık daire sayısı bulunan tüm planların eğitim datası olarak kullanıldığı algoritmanın sonucunda çıkan dokuz örnek plan şekil 4.24'de gösterilmiştir.



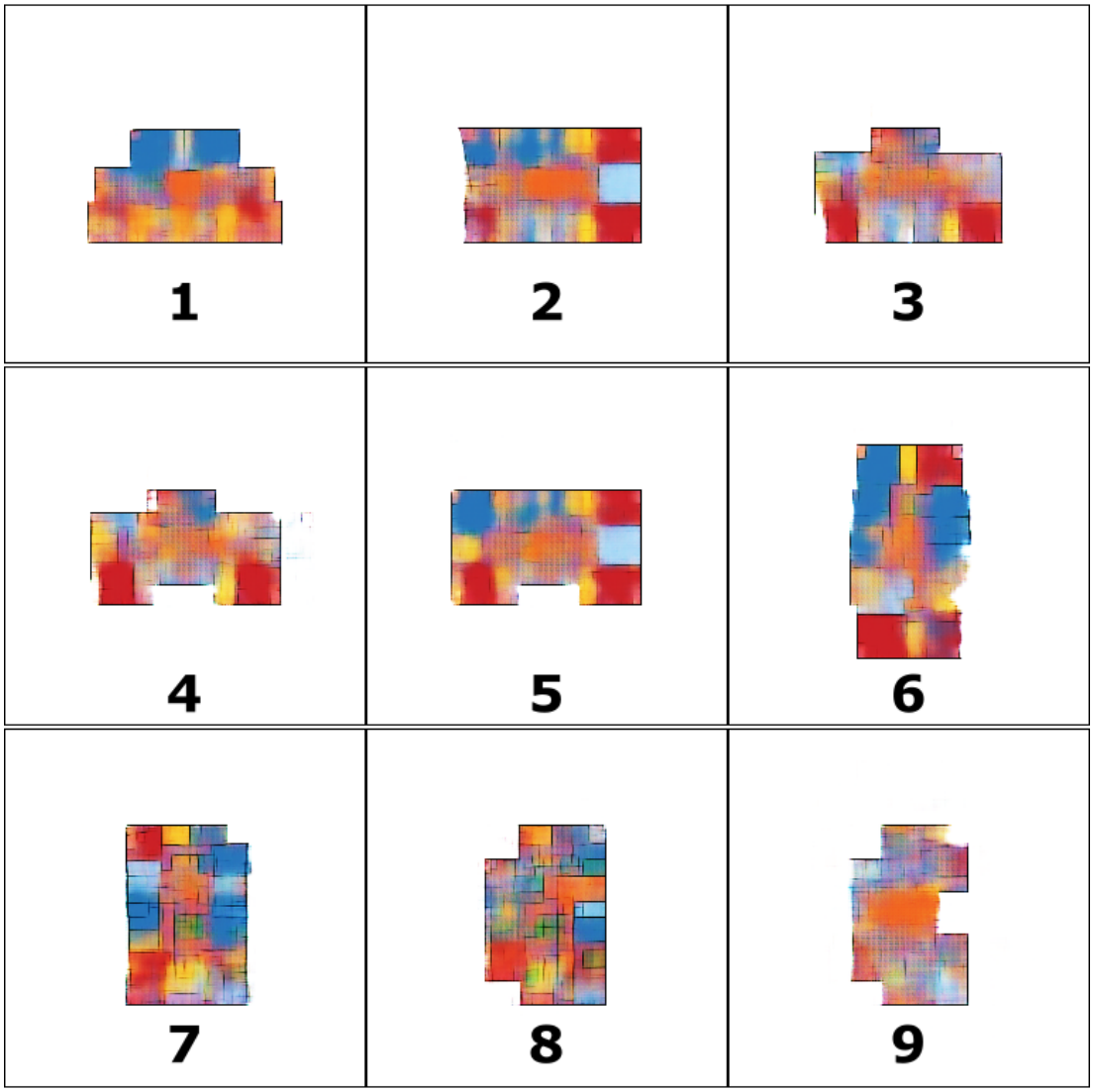
Şekil 4.24 d480 Devir Sayısı ve Tüm Planlar

Aşağıda d480 devir sayısı ile katta iki daire bulunan planlarının eğitim datası olarak kullanıldığı algoritmanın sonucunda çıkan dokuz örnek plan şekil 4.25'de gösterilmiştir.



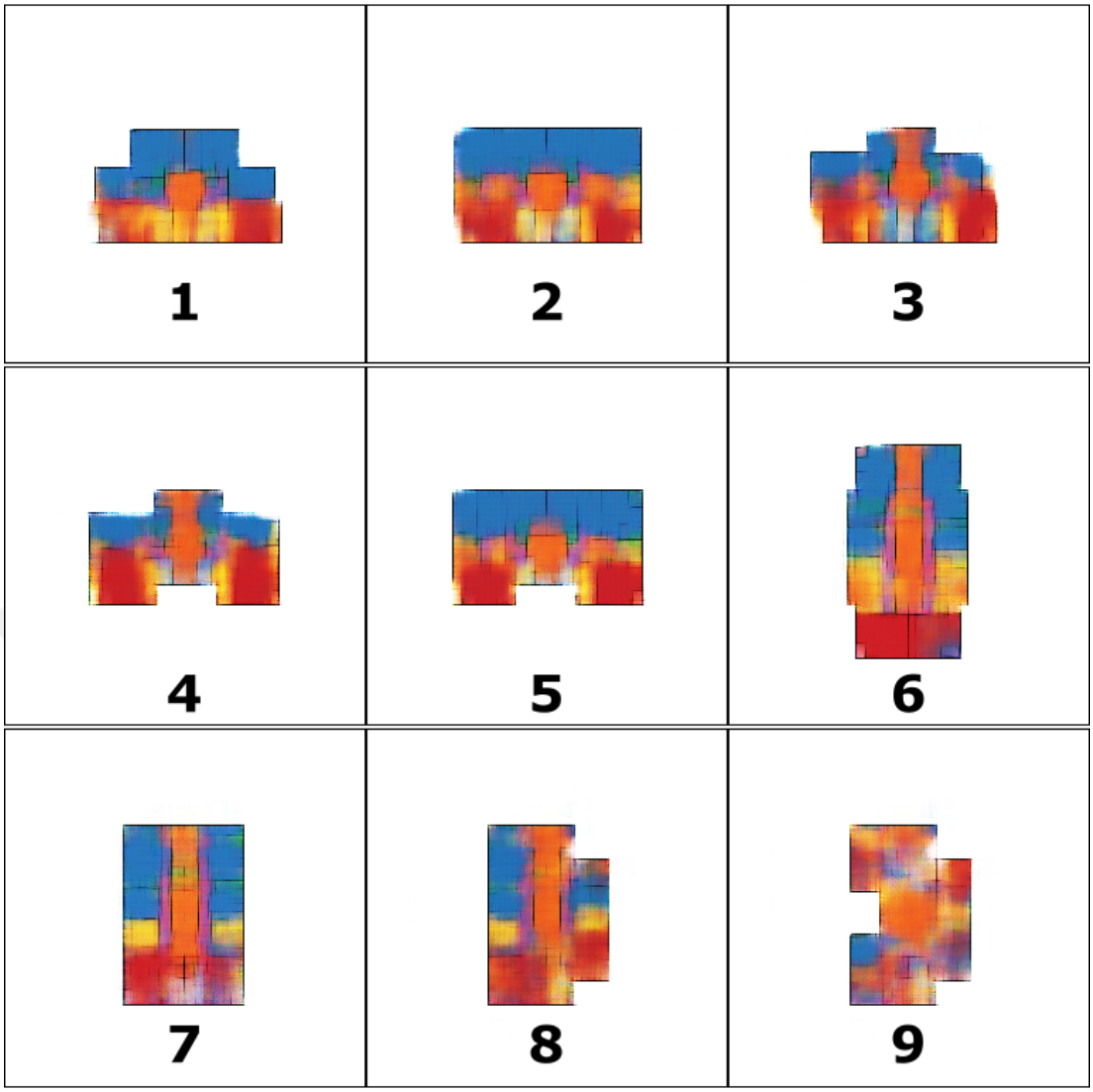
Şekil 4.25 d480 Devir Sayısı ve iki Daireli Planlar

Aşağıda d700 devir sayısı ile katta karışık daire sayısı bulunan tüm planların eğitim datası olarak kullanıldığı algoritmanın sonucunda çıkan dokuz örnek plan şekil 4.26'de gösterilmiştir.



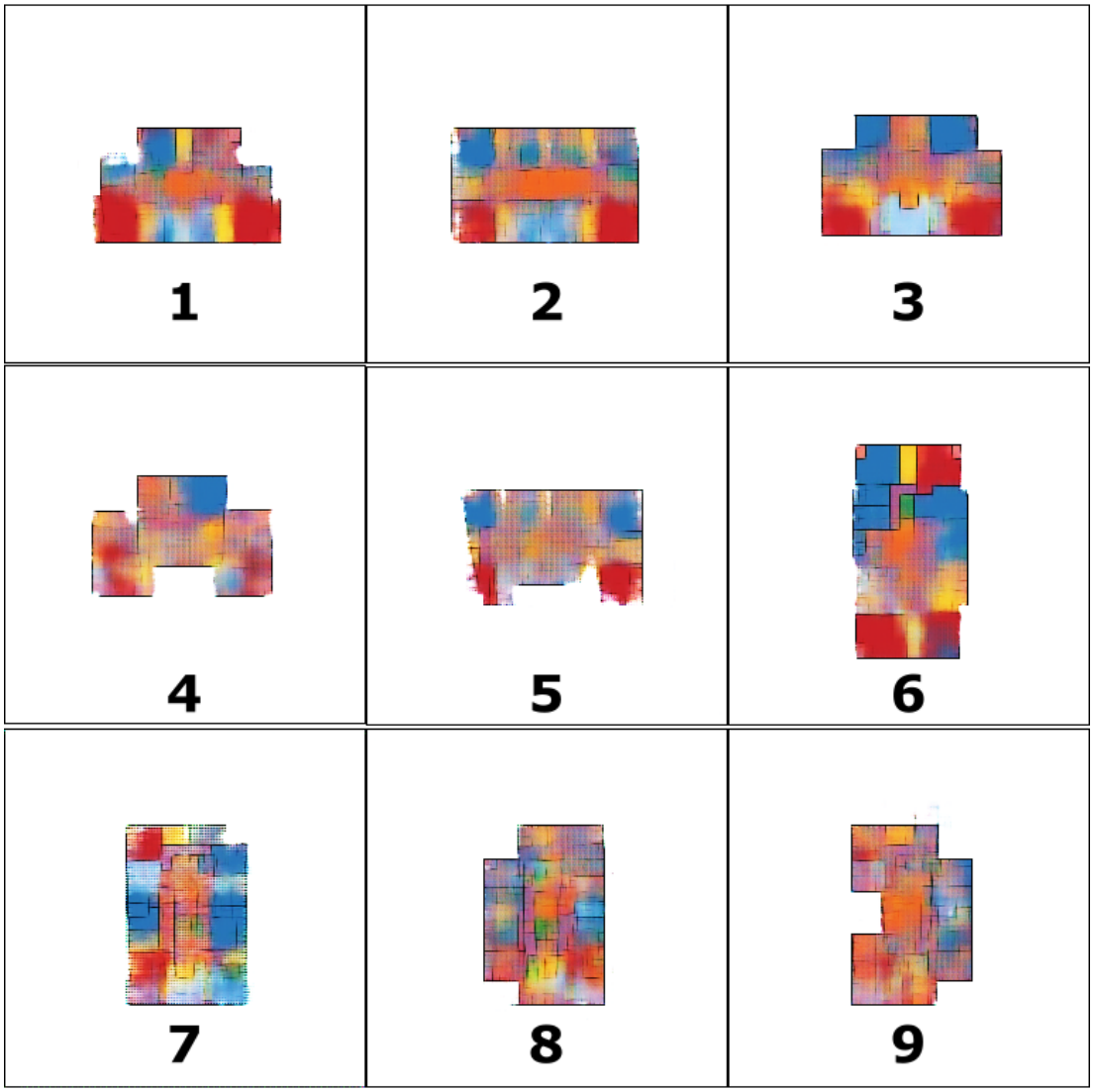
Şekil 4.26 d700 Devir Sayısı ve Tüm Planlar

Aşağıda d700 devir sayısı ile katta iki daire bulunan planların eğitim datası olarak kullanıldığı algoritmanın sonucunda çıkan dokuz örnek plan şekil 4.27'de gösterilmiştir.



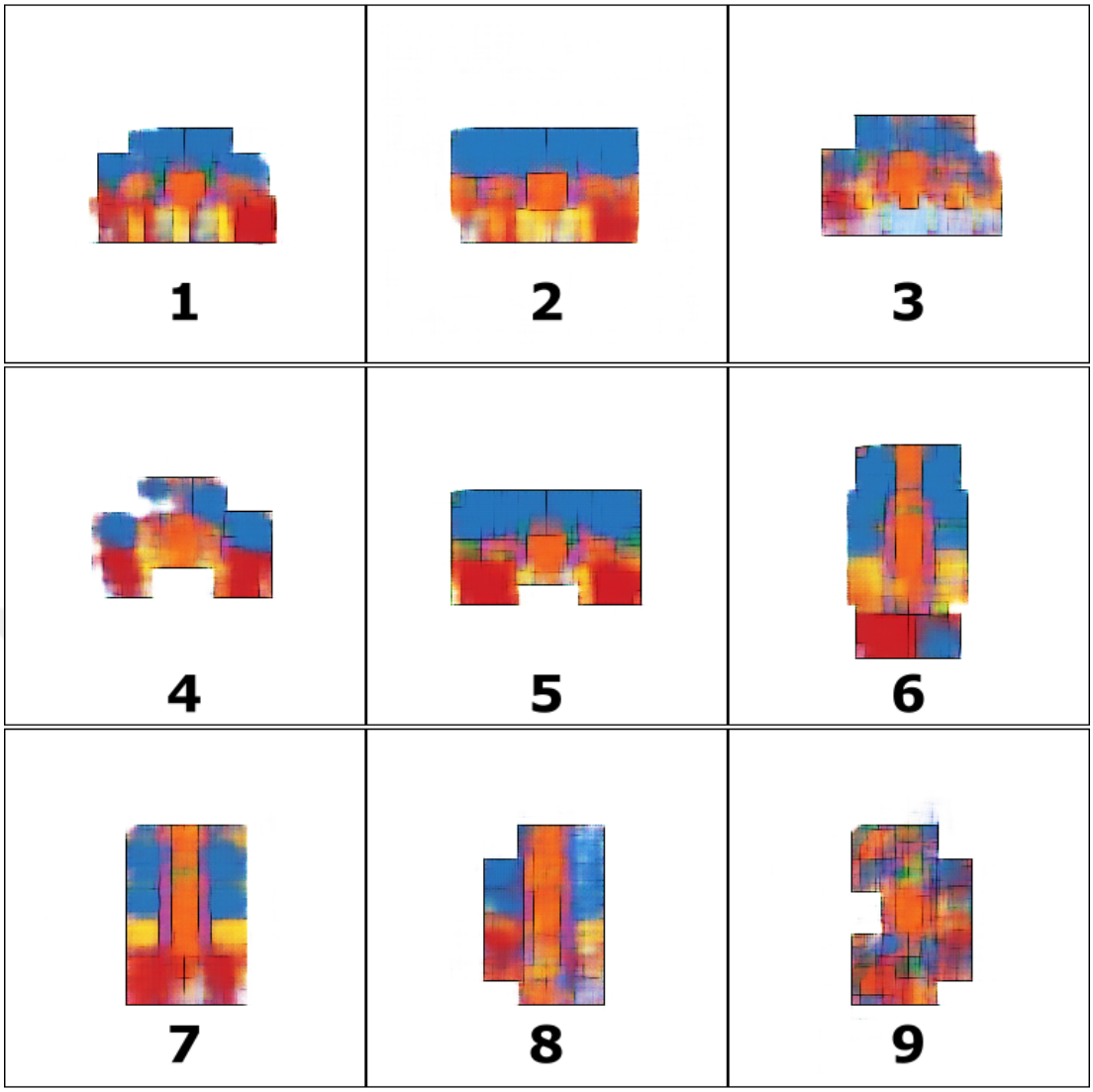
Şekil 4.27 d700 Devir Sayısı ve iki Daireli Planlar

Aşağıda d1000 devir sayısı ile katta karışık daire sayısı bulunan tüm planların eğitim datası olarak kullanıldığı algoritmanın sonucunda çıkan dokuz örnek plan şekil 4.28'de gösterilmiştir.



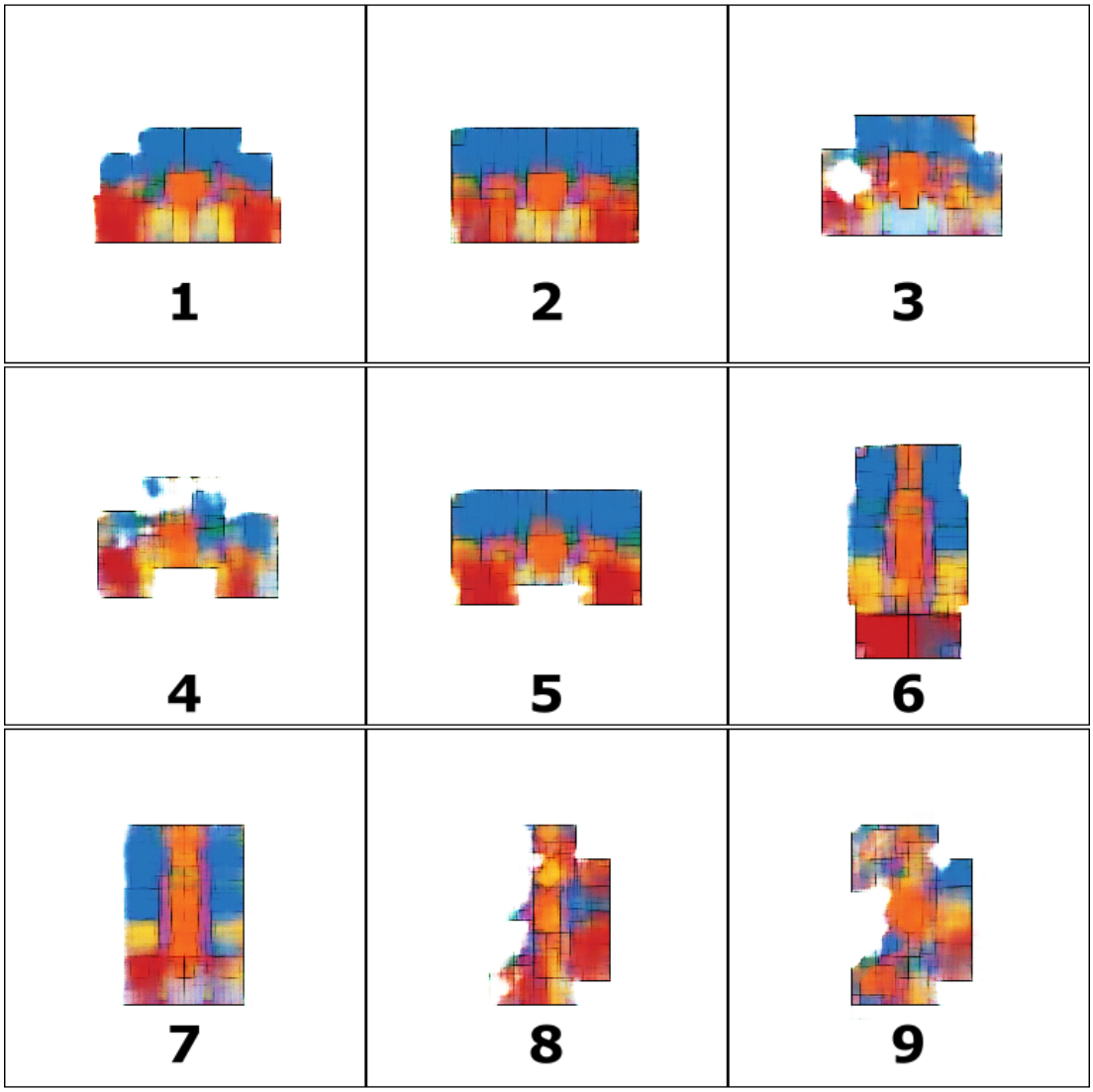
Şekil 4.28 d1000 Devir Sayısı ve Tüm Planlar

Aşağıda d1000 devir sayısı ile katta iki daire bulunan planların eğitim datası olarak kullanıldığı algoritmanın sonucunda çıkan dokuz örnek plan şekil 4.29'de gösterilmiştir.



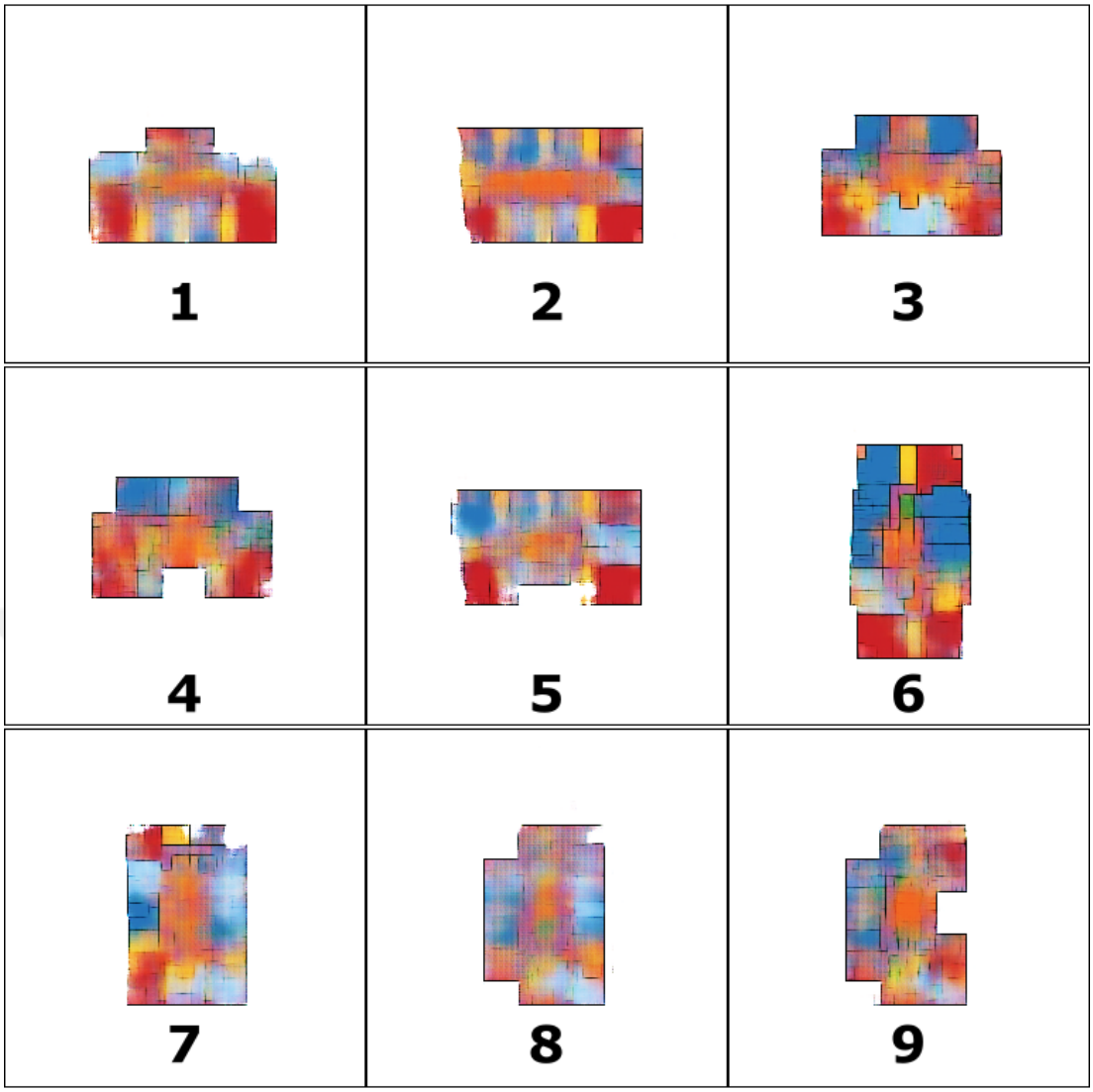
Şekil 4.29 d1000 Devir Sayısı ve iki Daireli Planlar

Aşağıda d1500 devir sayısı ile katta iki daire bulunan planların eğitim datası olarak kullanıldığı algoritmanın sonucunda çıkan dokuz örnek plan şekil 4.30'de gösterilmiştir.



Şekil 4.30 d1500 Devir Sayısı ve iki Daireli Planlar

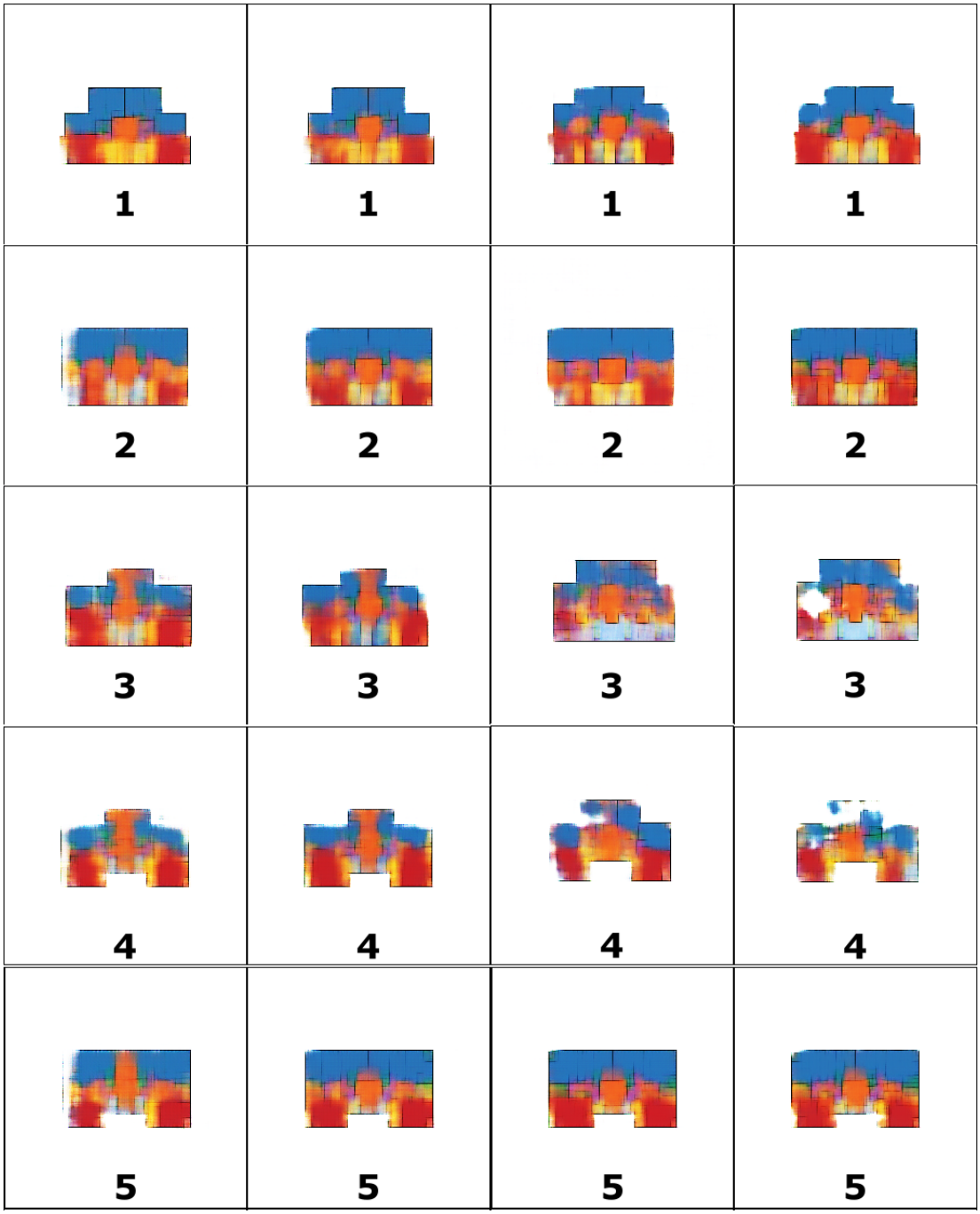
Aşağıda d1500 devir sayısı ile katta karışık daire sayısı bulunan tüm planların eğitim datası olarak kullanıldığı algoritmanın sonucunda çıkan dokuz örnek plan şekil 4.31'de gösterilmiştir.



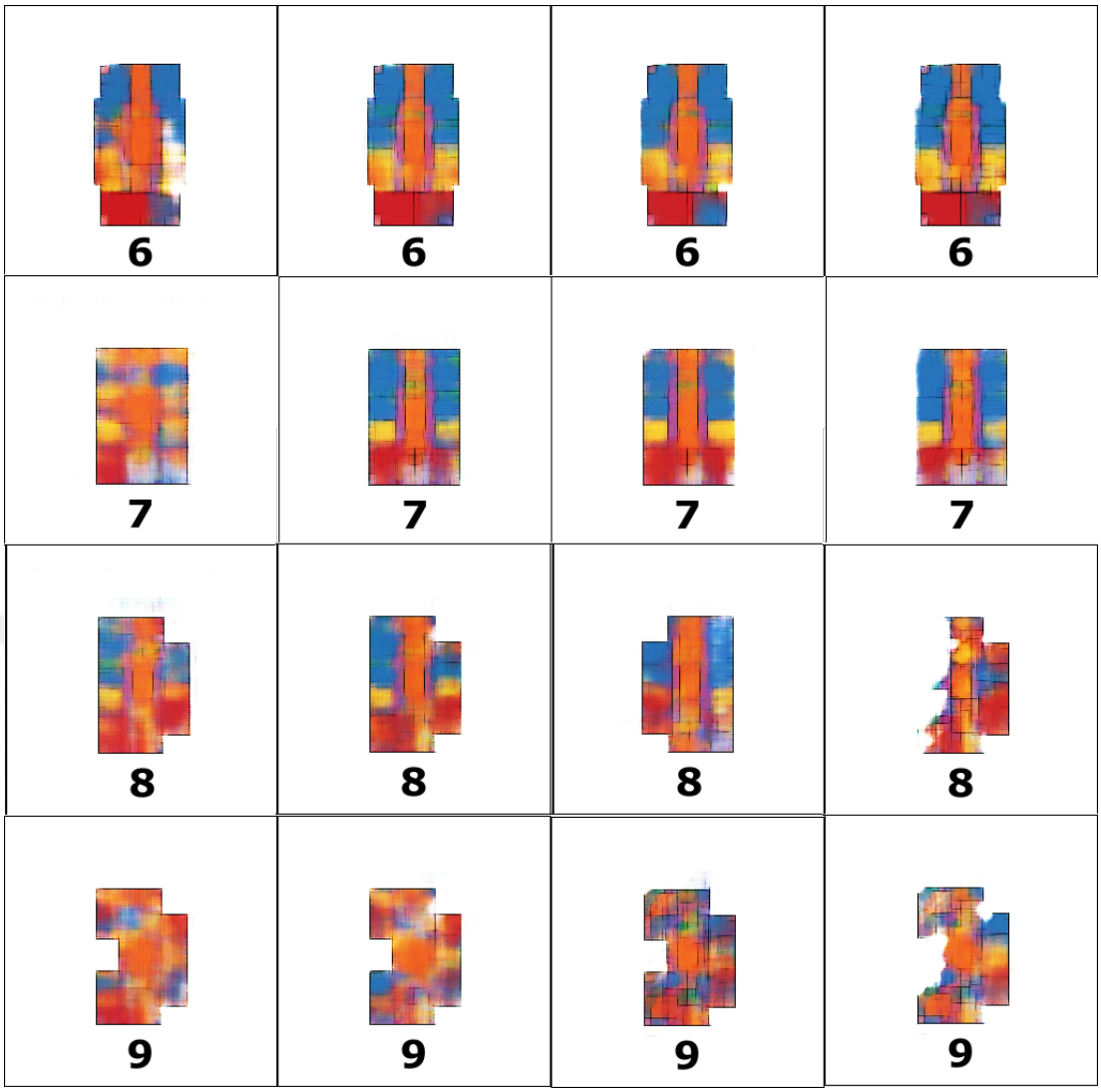
Şekil 4.31 d1500 Devir Sayısı ve Tüm Planlar

4.2.1 Hassasiyet Analizi

Planların doğruluğunun devir sayısı parametresine hassasiyetini ölçmek için model d480, d700, d1000 ve d1500 devir sayıları kullanılarak ayrı ayrı çalıştırılmış ve çalıştırılan eğitim verileri açısından en uygun devir sayıları olarak d700 ve d1000 değerleri tespit edilmiştir. Ayrıca, eğitim verisi olarak, belli bir kat sayısı için o kat sayısına uygun planların kullanılmasının uygun oldugu görülmüştür (Şekil 4.32, 4.33).



Şekil 4.32 Katta İki Daire Buluan Planlar ile Eğitilen Modellerin Ürettiği Örneklerin Devir Sayısına Göre Karşılaştırılması Soldan Sağa d480-d700-d1000-d1500

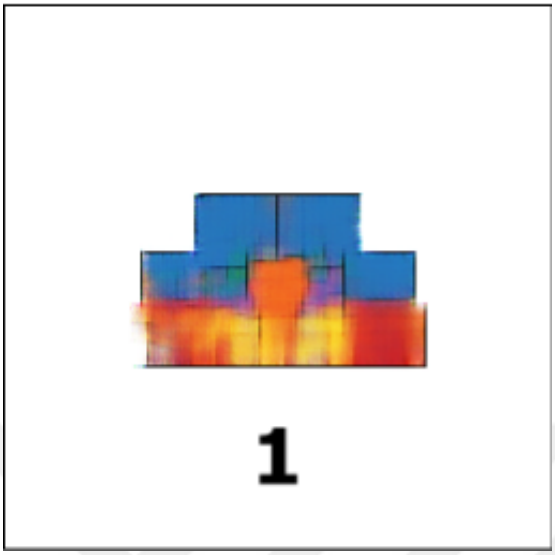


Şekil 4.33 Katta İki Daire Buluan Planlar ile Eğitilen Modellerin Ürettiği Örneklerin Devir Sayısına Göre Karşılaştırılması Soldan Sağa d480-d700-d1000-d1500

Devir sayısı parametresi d700'e ayarlanıp, iki daireli planların bulunduğu eğitim verileri ile model eğitildiğinde elde edilen 9 adet plana (Şekil 4.27) bakarak, modelin dogruluguna dair, odaların temsil ettiği renkler (Şekil 4.7) açısından şu gözlemler yapılmıştır:

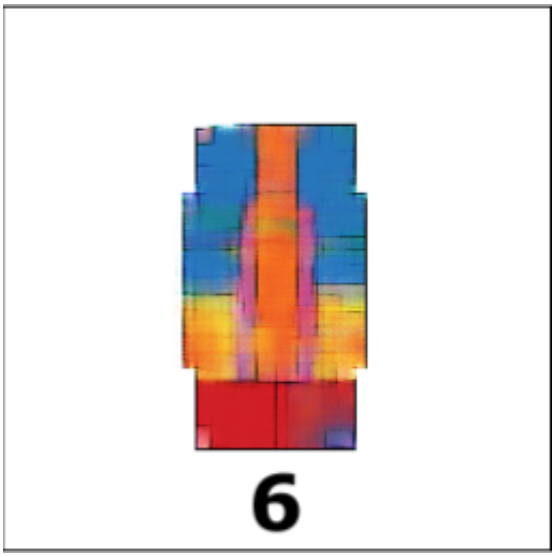
Üretici model, dış konturlerin netliği açısından yüksek doğrulukta planlar üretmiştir. Şekil 4.27'deki d700 devirli katta iki daireli planların ortalama doğruluk yüzdesi 93.5 olurken, en yüksek puanı alan 6 numaralı planda iki daire net olarak kat sirkulasyon alanı ile ayrılmış, her dairede gerekli odalar mimari kurgu açısından doğru biçimde yerleştirilmiştir.

Mavi renkle temsil edilen yatak odaları hemen hemen bütün planlarda cephelerle komşuluk halinde üretilmiştir. Üretici modelin bu alanları mimari planlama açısından tutarlı bir şekilde tasarladığı tespit edilmiştir (Şekil 4.34).



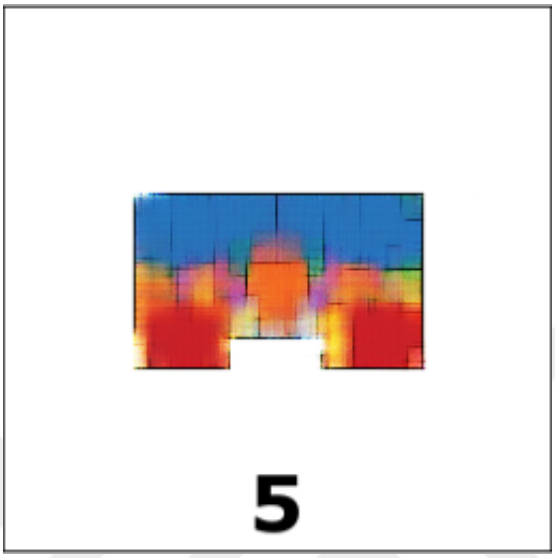
Şekil 4.34 d700 devirli iki daireli planlar, 1 numaralı örnek

Pembe renkle üretilen balkon alanlarının bazı planlarda tasarlandığı bazılarında ise tasarlanmadığı görülmüştür. Üretici modele verilen girdi görselinin geometrisine ve eğitim verilerinde bu girdi görseline benzer planlarda balkonun olup olmamasına göre, model bu pembe renkli mahalleri oluşturmuş veya oluşturmamıştır. Bu mahaller model tarafından genellikle köşelerde ve cephelere komşu olacak şekilde yerleştirilmiştir ki bu mimari kurgu açısından uygun görülmüştür (Şekil 4.35).



Şekil 4.35 d700 devirli iki daireli planlar, 6 numaralı örnek

Kırmızı renkle gösterilen salon mahalleri üretici modelin tasarladığı çoğu örnekte mimari kurgu bakımından doğru lokasyona yerleşmiştir.Genelgeçer mimari kurguya uygun olarak planlarda daha geniş yer kaplamış ve dikdörtgen veya kare gibi düzgün geometrik alanlarla temsil edilmiştir. Ayrıca, odalar net çizgiler ile ayrılarak oda geometrisi belirginleşmiştir (Şekil 4.36).



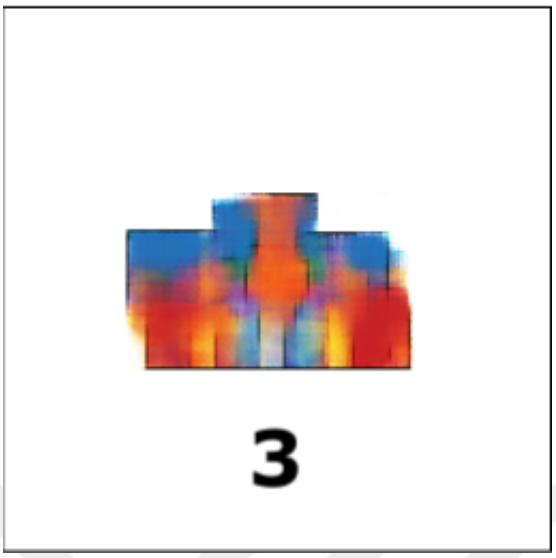
Şekil 4.36 d700 devirli iki daireli planlar, 5 numaralı örnek

Mor renkle gösterilen hol alanları lokasyon olarak genellikle tüm odalara komşu, cepheden ayrı ve karanlık alanlarda kalarak yerleşimleri mimari kurgu açısından tutarlıdır. Ayrıca, bu alanların geometrisinin dar ve uzun olması yine genelgeçer yapı mimarisine uygun görülmüştür (Şekil 4.35).

Sarı renk ile gösterilen mutfak alanları plan içerisinde genellikle salona yakın bir şekilde konumlanmış, bunların geometrisi çoğu zaman diğer renklerle karışarak belirginliğini kaybetmiştir (Şekil 4.35). Bu durum, modelin büyük alanlara nazaran mutfak gibi küçük alanları ayırt etmekte zorlandığına işaret etmektedir. Ancak, modelin yeterli sayıda küçük alanların yer aldığı yeterli sayıda çalıştırılması halinde bu sorunun ortadan kalkacağı düşünülebilir. Bu türden bir çalışma gelecekteki başka araştırmalarda ele alınabilir.

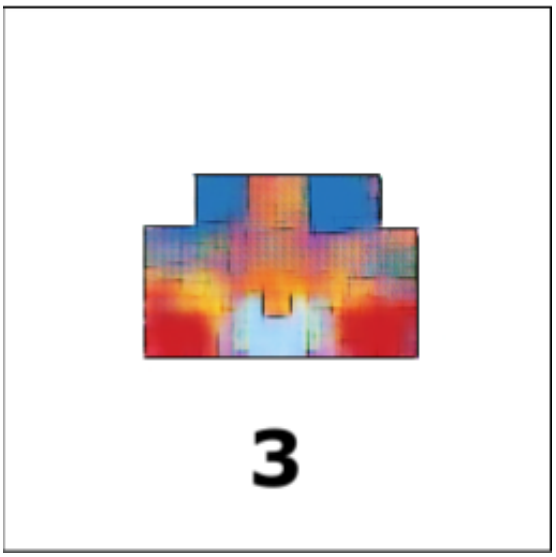
Yeşil renk ile gösterilen ıslak hacim alanlarının modelin ürettiği planlarda kapladığı alanın büyüklüğü genelgeçer mimari kabullere uygundur. Ancak, mutfak alanlarındaki soruna benzer bir biçimde, bu alanlar diğer alanlarla kısmen iç içe geçerek geometrik netliğini yitirmiştir. Model, ıslak alanları yatak odası gibi yaşam alanlarına

yakın ve karanlık alanlarda olmak üzere mimari kurguya uygun olarak konumlandırmıştır (Şekil 4.37).



Şekil 4.37 d700 devirli iki daireli planlar, 3 numaralı örnek

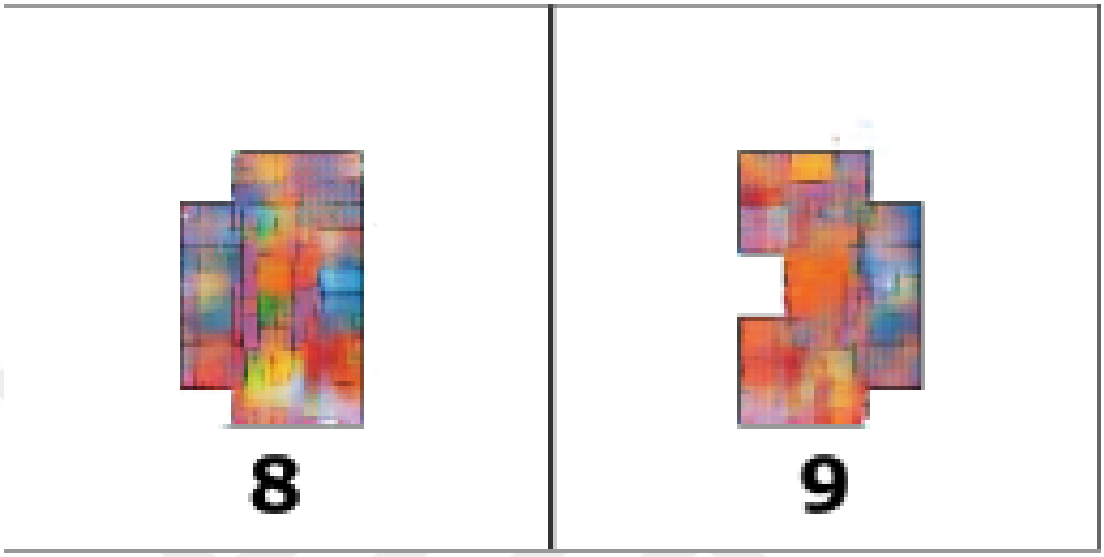
Açık mavi ile gösterilen renk teras alanlarını temsil etmektedir. Üretici modele verilen girdi planın geometrisine ve eğitim verilerinde bulunan benzer planların terasa sahip olma durumuna göre model bunları genel olarak dogru yerleştirmiştir. Ancak, netlik sorunu bu mahaller açısından da söz konusudur (Şekil 4.38).



Şekil 4.38 d1000 devirli karışık daire sayılı planlar, 3 numaralı örnek

Genel olarak model, dikey olarak simetrik olmayan girdi planlarında çok düşük bir doğruluk oranına ulaşmıstır (Şekil 4.39. Bu, eğitim verileri arasinda yeterli sayıda

dikey-simetriye sahip plan bulunmamasından kaynaklanmıştır. Bütün girdi verileri arasinda dikey-simetrik plan sayisi %27'dir. Genel olarak, üretilen planların kalitesinin artırılması için istenen tipte planların cephe geometrisine uygun planların modelin eğitim sürecinde kullanılması gerekmektedir.



Şekil 4.39 d1000 devirli karışık daire sayılı planlar, 8 ve 9 numaralı örnek

4.2.2 Ortalama Kare Hatası / Kök Ortalama Kare Hatası Yöntemi ile Model Performans Analizi

Ortalama kare hatası (MSE) modelin ürettiği tahmini verilerin gerçek verilerden ne kadar farklı olduğuna dair mutlak bir sayı verir. Her pikselin hatasını gerçek veri ile tahmini veri arasında analiz ederek sonuçta 0 ile 1 arasında bir sayı verir. 0 puanı hatasız 1 ise tamamen hatalı anlamına gelmektedir. Hesaplama yöntemi formül (4.1)'de gösterildiği gibidir. Kök ortalama kare hatası (RMSE), MSE'nin karekökü̈dür. Hesaplama yöntemi formül (4.2)'de gösterildiği gibidir. MSE değeri kolayca karşılaştırılamayacak kadar büyük olabilir, bu durumlarda RMSE kullanılır. Bu yüzden MSE hata karesi ile hesaplanır ve yorumlamayı kolaylaştırır.

Ortalama Kare Hatasını python dilinde yazılmış "from sklearn.metrics import mean_squared_error" kütüphanesini kullanılarak hesaplanmıştır. Bu Fonskiyonun ilk parametresine gerçek veriler ikinci parametresine ise modelin tahmin ettiği değerler verilir. Kök Ortalama Kare Hatası'nı hesaplarken ise karekök almak için numpy kütüphanesinin sqrt fonksiyonu kullanılmıştır.

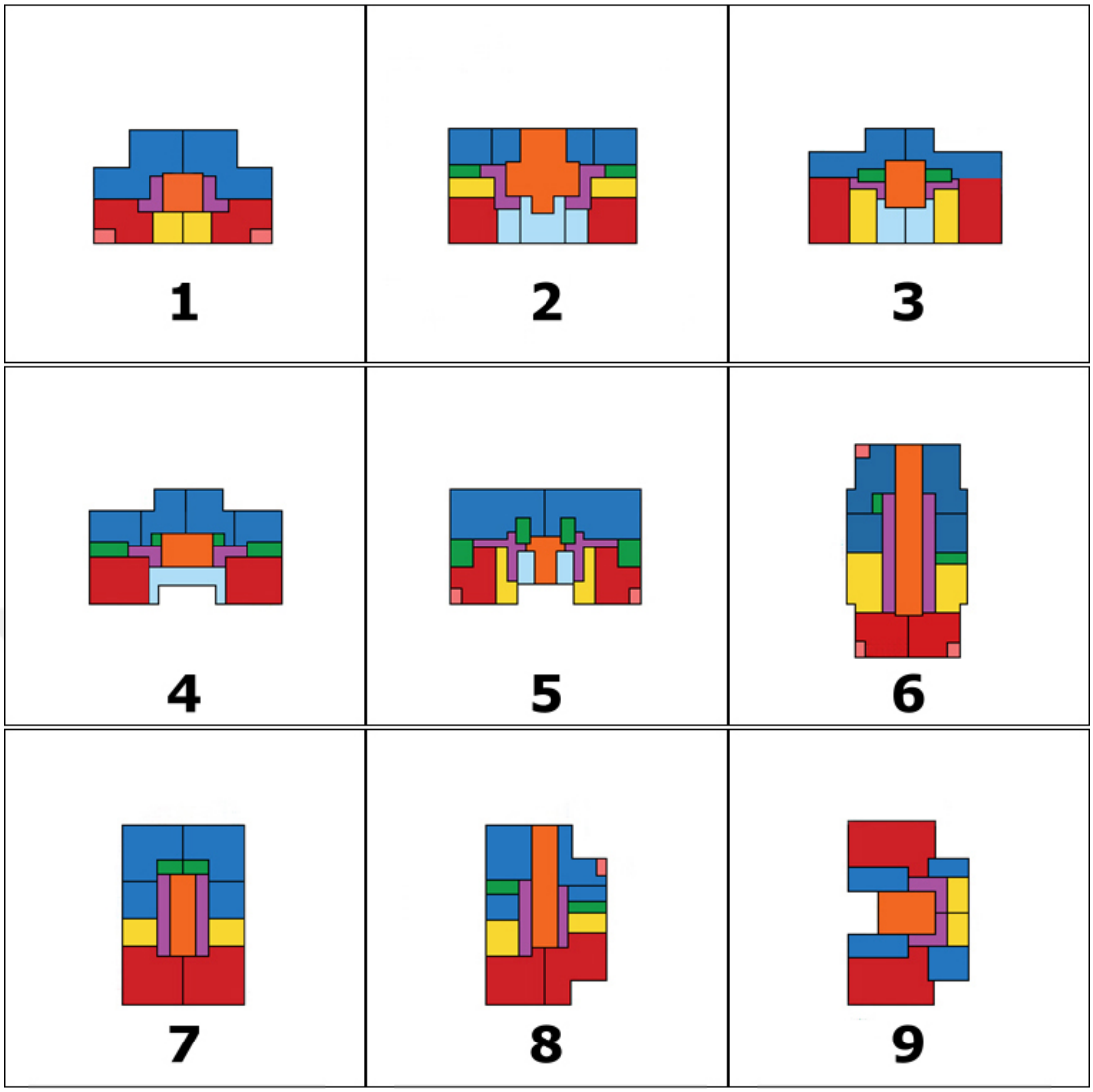
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad (4.1)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \quad (4.2)$$

Algoritmanın kaynak kodları Ek A.11'de gösterilmiştir. RMSE fonksiyonunun çalışması için gerekli parametreleri ve değerlerini gösteren gereken kod aşağıdaki gibidir (nekhtiari, 2020).

image-similarity-measures –org_img_path= *gerçek verinin konumu* –pred_img_path= *tahmini verinin konumu* –metric rmse

Katta iki daireli planlar ile eğitilmiş modellerden alınan verilen RMSE ile analiz edilmiştir. Bu analizi gerçekleştirmek için gerçek verilere ihtiyaç vardır. Modelden alınan veriler tamemen tahmini veri olduğu için mevcutta gerçek veriler bulunmamaktadır. Modelin ürettiği farklı devir sayılarında ürettiği örnek dokuz veri göz önünde bulunarak, olabilecek en doğru çözüm görseli oluşturulmuştur. Bu görseller Şekil 4.40'de bulunmaktadır.



Şekil 4.40 RMSE için tahmini gerçek veriler

Buna göre farklı devir sayılarına göre eğitilen modellerin ürettiği görseller (Şekil 4.32 & Şekil 4.33) ile gerçek görseller 4.40 arasındaki hatayı hesaplayan RMSE algoritmasından alınan sonuçlar aşağıdaki gibidir.

#	d480	d700	d1000	d1500
1	0.005246726330369711	0.005470018833875656	0.007767609786242247	0.007552749011665583
2	0.010739660821855068	0.010421033017337322	0.010295141488313675	0.009925007820129395
3	0.00871185027062893	0.00769776152446866	0.011054016649723053	0.011477532796561718
4	0.009461604058742523	0.008737951517105103	0.010657225735485554	0.010830283164978027
5	0.011138026602566242	0.010066908784210682	0.007825912907719612	0.00804134365171194
6	0.007886857725679874	0.006472766399383545	0.007270692382007837	0.007088674698024988
7	0.01016567088663578	0.0067746601998806	0.00719974422827363	0.007977813482284546
8	0.010540013201534748	0.008863192982971668	0.01122922170907259	0.012549754232168198
9	0.007998109795153141	0.008295931853353977	0.008431950584053993	0.009755396284162998

Tablo 4.1 Örnek Görseller İçin RMSE Sonuçları

Bu sonuçlara göre:

- Bu dokuz görsel için en düşük RMSE ortalaması ile en tutarlı model d700 daha sonra sırasıyla d1000, d480 ve d1500 olmuştur. RMSE ortalamalar ise: d700 = 0.00808891390139856, d1000 = 0.00908127949676577, d480 = 0.00909872441035175, d1500 = 0.00946650612685413'dir.
- 1. görsel için en tutarlı çözüm d480 daha sonra d700,d1500,d1000 devir sayısı ile eğitilen modellerden üretilmiştir.
- 2. görsel için en tutarlı çözüm d1500 daha sonra d1000,d700,d480 devir sayısı ile eğitilen modellerden üretilmiştir.
- 3. görsel için en tutarlı çözüm d700 daha sonra d480,d1000,d1500 devir sayısı ile eğitilen modellerden üretilmiştir.
- 4. görsel için en tutarlı çözüm d480 daha sonra d480,d1000,d1500 devir sayısı ile eğitilen modellerden üretilmiştir.
- 5. görsel için en tutarlı çözüm d1000 daha sonra d1500,d700,d480 devir sayısı ile eğitilen modellerden üretilmiştir.
- 6. görsel için en tutarlı çözüm d700 daha sonra d1500,d1000,d480 devir sayısı ile eğitilen modellerden üretilmiştir.
- 7. görsel için en tutarlı çözüm d700 daha sonra d1000,d1500,d480 devir sayısı ile eğitilen modellerden üretilmiştir.

- 8. görsel için en tutarlı çözüm d700 daha sonra d480,d1000,d1500 devir sayısı ile eğitilen modellerden üretilmiştir.
- 9. görsel için en tutarlı çözüm d700 daha sonra d480,d1000,d1500 devir sayısı ile eğitilen modellerden üretilmiştir.
- Örnek görseller arasında en tutarlı model çıktısı 1. Görsel için d480 devir sayısı ile eğitilen model tarafından üretilmiştir.
- Örnek görseller arasında en hatalı model çıktısı 8. Görsel için d1500 devir sayısı ile eğitilen model tarafından üretilmiştir.

5. SONUÇ VE ÖNERİLER

Şematik plan tasarlama metotlarının, güncel bilgi birikimi, teknoloji ve yöntemler ile yeniden çözümlenebileceği ve yeni yöntemler geliştirilebileceği görülmektedir. ÇÜA algoritmaları da güncel teknoloji ve bilimin, bu amaç için çözüm sunduğu en yeni yöntemlerden biridir. Bu çalışma sonucunda, ÇÜA algoritmalarından pix2pix mimarisini kullanarak belirli bir alan sınırları içinde bölgesel standartlara ve yönetmeliklere uygun eğitim verileri kullanılarak şematik plan üretimi yapılmıştır. Şematik plan üretiminde kullanılan verilerin model eğitimi için gerekli nitelikleri ve nicelikleri karşılaştırılmıştır. Model eğitiminde kullanılan eğitim verilerinin sahip olması gereken özellikleri belirtilmiştir. Model önce algoritmik olarak üretilmiş yapay veriler kullanılarak test edilmiş, başarılı sonuçlar alındıktan sonra gerçek verilere uyarlanmıştır. Ayrıca bu eğitim verileri ile eğitilecek modellerin eğitim parametrelerinden olan devir sayısına hassasiyeti görsel sonuçlar eşliğinde analiz edilmiştir. Model önce elde bulunan kattaki daire sayılarını gözetmeksizin bütün plan içerikleri kullanılarak çalıştırılmış, ardından planlar kategorize edilmiş, bu kategoriler içinde yeterli sayıda veri bulunan iki katli planlar için model ayrica test edilmiştir.

Eğitim verileri belirli bir mimari kurgu için kategorize edildiğinde, her bir kategoriye yönelik olarak çalıştırılan pix2pix algoritmalarından daha tutarlı sonuçlar elde edileceği görülmektedir. Bu çalışmada iki daireli kat planları ve farklı sayıda daireli kat planları kullanılarak eğitilen modellerin ürettiği çıktılar karşılaştırılmıştır. Mimari olarak iki daireli planların daha tutarlı sonuçlar verdiği tespit edilmiştir. Modellerin devir sayısında yapılan hassasiyet analizi sonucunda aşırı eğitilmiş veya yeteri kadar eğitilmemiş modellerin eksik veya düşük kaliteli sonuçlar verdiği, optimal değerde eğitilen modellerin ise daha tutarlı planlar ürettiği tespit edilmiştir. RMSE sonuçlarına göre d1500 hariç diğer devir sayılarında eğitilmiş modellerinde en az bir görsel için en tutarlı puanı aldığı gözlemlenmiştir. Yine RMSE sonuçlarına göre en tutarlı görselin üretildiği model ile genelde en tutarlı görselleri üreten modelin devir sayıları birbirinden farklıdır. Bu sonuçlar göz önüne alındığında eğitim modelinde bulunan kat

planlarının geometrik şekilleri optimal devir sayısına etki ettiği anlaşılmaktadır. Bir eğitim seti ile eğitilmiş modelden alınan farklı girdiler için bir optimal devir sayısının olamayacağı verilen girdi görseline bağlı olarak her plan için ayrı bir optimal devir sayısı olacabileceği anlaşılmaktadır. Böylelikle şematik plan üretme iş akışında farklı devir sayıları ile üretilmiş çoklu modeller kullanılabilir ve bunlar arasında girdi görsellerine göre seçim yapılabilir. Bu seçimi yaparken modellerin birbirleri arasında RMSE sonuçlarına göre en tutarlı puanı alan modelin ürettiği görseller kullanılabilir veya RMSE kullanmadan göze en doğru gözüken planlar da kullanılabilir.

Bu türden gelişmiş yapay zeka algoritmalar sayesinde mimarlık veya plan çözümü bilgisi olmayan kişilerin de parsel, yüzey veya dış konturları belirli alanlar için mimari şematik plan çözümlerine hızlı biçimde erişebilecekleri açıktır. Bu planlar, mimari bilgisi olan kişilere dahi alternatif çözümler olarak hizmet sunabilir. Bu çalışmanın amaçlarından biri olan yapay zeka teknolojilerinin mimari alanda kullanımına yönelik olanakların tespit edilmesi konusunda ÇÜA'nın hızlı plan üretme konusunda yol açıcı olduğu kanısına varılmıştır. Kent mimarisi sorumluluğunda olan kurumlar, ellerinde bulunan mevcut planları eğitim verileri olmak üzere yapay zeka algoritmalarını kullanan araştırmacıların, mimarların ve şehir planlamacıların hizmetine sunarak, birbirine alternatif planların hızlı bir biçimde üretilip tartışmaya açılmasına vesile olabilir.

Bu türden algoritmaların araştırma düzeyinde kalmaması, işlevlik kazanması ve gerçek hayatta uygulamalar bulabilmesi için, karar vericilerin bu türden çalışmaları desteklemesi ve özellikle kentsel yapı verilerini elinde bulunduran kurumların veri bankalarını bu araştırmaları yürüten tarafların hizmetine sunması elzemdir. Bu veri bankalarındaki planların, yapı özelliklerine, yönetmeliklere ve standartlara uygunluğuna göre etiketlenmesi ve böylelikle dijital ortamdan erişimlerinin kolaylaştırılması, yapay zeka uygulamalarının çok hızlı bir biçimde gelişmesine hizmet edecektir. Örneğin, bu çalışma kapsamında nispeten basit eğitim verileri kullanılarak üretilen planlar şematik plan tasarımı için fikir verici ön tasarım planları sunmakla beraber, bu planların son kullanıcı için yeterli düzeyde bir işlevliği bulunmamaktadır. Ancak, daha geniş bir veri tabanı ile eğitilecek benzer yapay zeka algoritmalarından gerçek hayatta uygulama bulabilecek çok daha kapsamlı planların üretebileceği açıktır.

Bu çalışma, yapay zeka algoritmalarının temel düzeydeki iki boyutlu mimari plan-

lar üretilmesi konusundaki potansiyelini araştırmaktadır. Kısıtlı sayıdaki eğitim verileri kullanıldığında dahi, uygun parametrelerle eğitilen modellerden tatminkar sonuçlar elde edilebileceği görülmüştür. Geniş bir çeşitlilikte eğitim verileri ve amaca uygun farklı yapay zeka uygulamaları kullanılarak gerçek hayatta karşılık bulabilecek üç boyutlu planların üretilmesi ileriki çalışmaların konusu olacaktır. Bu çalışma ile, yapay zeka algoritmalarının, tasarım aşamaşından inşa aşamasına kadarki bütün süreçlerde devrimci nitelikte yenilikler getirebilecek olanaklar sunacağı inancı gelişmiştir. Ayrıca ileriki çalışmalarda çok daha kapsamlı eğitim verileri kullanılarak, çok çeşitli cephe geometrileri için, iki ve üç boyutlu tasarımlar üretmek üzere, pix2pix ile birlikte farklı yapay zeka algoritmaları test edilebilir.

KAYNAKÇA

Ahmed, S., Liwicki, M., Weber, M., & Dengel, A. (2012). Automatic room detection and room labeling from architectural floor plans. *2012 10th IAPR interna- tional workshop on document analysis systems*, 339–343.

Akdoğan, E. (2017). *Yapay sinir ağlarına giriş*. http://ytubiomechatronics.com/wp-content/uploads/2017/10/YSA.pdf

Akküç, G. (2019). *Ulusal yapı enformasyon modellemesi (yem/bim) kütüphanesi içer- iğinin sektörel ihtiyaçları karşılayacak şekilde geliştirilmesi için bir araştırma: Kapı örneği* (Master's thesis). İstanbul Teknik Üniversitesi.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6.

Ansari, S. (2022). *Pattern recognition | introduction*. https://www.geeksforgeeks.org/ pattern-recognition-introduction/

As, I., Pal, S., & Basu, P. (2018). Artificial intelligence in architecture: Generating conceptual design via deep learning. *International Journal of Architectural Computing*, *16*(4), 306–327.

Ataseven, B. (2013). Yapay sinir ağlari ile öngörü modellemesi. *Öneri Dergisi*, *10*(39), 101–115.

Bengio, Y., Goodfellow, I., & Courville, A. (2017). *Deep learning* (Vol. 1). MIT press Massachusetts, USA:

Bilgin, M. (2017). Gerçek veri setlerinde klasik makine öğrenmesi yöntemlerinin performans analizi. *Breast*, *2*(9), 683.

Caetano, I., Santos, L., & Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. *Frontiers of Architectural Research*, *9*(2), 287–300.

Chaillou, S. (2019). Ai+ architecture: Towards a new approach. *Harvard University*, *188*.

Dick, S. (2019). Artificial intelligence.

Engin, O. (2001). *Akış tipi çizelgeleme problemlerinin genetik algoritma ile çözüm performansının artırılmasında parametre optimizasyonu* (Doctoral dissertation). Fen Bilimleri Enstitüsü.

Erbaş, S. K. (2015). Hesaplamalı tasarım yaklaşımının bilgisayar destekli tasarım eğitimine etkisi.

Fogg, A. (2018). *A history of machine learning and deep learning*. https://www.import.io/post/history-of-deep-learning/

Fordjour, H. A. (2019). *<https://blog.paperspace.com/unpaired-image-to-image- translation-with-cyclegan/>*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Gündoğdu, M. (2014). Mekan dizimi analiz yöntemi ve araştırma konuları (space syn- tax and researching issues). *Art-Sanat Dergisi*, (2), 251–274.

Hillier, B., & Hanson, J. (1997). The reasoning art. *The 1st International Space Syntax Symposium Proceedings*.

Hindupur, A. (2018). *<https://deephunt.in/the-gan-zoo-79597dc8c347>* erişim tarihi 2022/08/12.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tuto- rial. *Computer*, *29*(3), 31–44.

Kahng, M., Thorat, N., Chau, D. H., Viégas, F. B., & Wattenberg, M. (2018). Gan lab: Understanding complex deep generative models using interactive visual experimentation. *IEEE transactions on visualization and computer graphics*, *25*(1), 310–320.

Kahng, M., Thorat, N., Chau, D. H., Viégas, F. B., & Wattenberg, M. (2019). *<https://poloclub.github.io/ganlab/>* erişim tarihi 2021-11-18.

Kalay, Y. E. (1985). Redefining the role of computers in architecture: From drafting/modelling tools to knowledge-based design assistants. *Computer-Aided Design*, *17*(7), 319–328.

Kalay, Y. E. (1989). Modeling objects and environments.

Kalay, Y. E. (1999). The future of caad: From computer-aided design to computer-aided collaboration. *Computers in building* (pp. 13–30). Springer.

Kolarevic, B. (2003). Digital morphogenesis. *Architecture in the digital age: Design and manufacturing*, 12–28.

Koray, C. (2016). *Bilgisayarla görü tabanlı satranç oyunu izleme sistemi* (Master's thesis). Başkent Üniversitesi Fen Bilimleri Enstitüsü.

Kubat, A. S., İnce Güney, Y., & Özer, Ö. (2007). *The machine learning future is now: How ai is disrupting entire industries*. https://v3.arkitera.com/h15866-space- syntax-uzerine.html

lila Mimarlık. (2021). *Oğuzhan umur bozkurt & gökçe uysalkan bozkurt firma arşivi* [İstanbul].

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, *9*, 381–386.

Merrell, P., Schkufza, E., & Koltun, V. (2010). Computer-generated residential build- ing layouts. *Acm s□ggraph asia 2010 papers* (pp. 1–12).

Mitchell, T. M. (1997). *Machine learning* (Vol. 92). McGraw-Hill, New York:

Moretti, L. (1971). Ricerca matematica in architettura e urbanistica.

Nekhtiari. (2020). *Https://github.com/up42/image-similarity-measures.*

NG, A. (2018). *<https://www.coursera.org/learn/machine-learning>* erişim tarihi 2020/06/18.

NG, A. (2019). https://www.coursera.org/learn/neural-networks-deep-learning erişim tarihi 2021/08/18

NIBS. (2007). *National building information modeling standard v1 part 1 overview, principles, and methodologies.*

Nourian, P., Rezvani, S., & Sariyildiz, S. (2013). Designing with space syntax.

Ofluoğlu, S. (2014). Yapı bilgi modelleme: Gereksinim ve birlikte çalışabilirlik. *Mimarist, Ocak.*

Priddy, K. L., & Keller, P. E. (2005). *Artificial neural networks: An introduction* (Vol. 68). SPIE press.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learn- ing with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229.

Sutherland, I. E. (1964). Sketchpad a man-machine graphical communication system. *Simulation*, *2*(5), R–3.

Szalapaj, P. (2013). *Cad principles for architectural design*. Routledge.

Tüfekçi, M., & Karpat, F. (2019). Derin öğrenme mimarilerinden konvolüsyonel sinir ağları (cnn) üzerinde görüntü işleme-sınıflandırma kabiliyetininin arttırılmasına yönelik yapılan çalışmaların incelenmesi. *International Conference on Human-Computer Interaction, Optimization and Robotic Applications*, 28–31.

Url-01. *<https://pytorch.org/docs/stable/index.html>* erişim tarihi 2021/11/18.

Url-02. *<https://ecbsite.com/de/2018/04/11/9-artificial-intelligence-ai-trends-most-commonly-used/>* erişim tarihi 2020/05/18.

Url-03. <*https://history-computer.com/sketchpad-guide/*> erişim tarihi 2022/01/23.

Url-04. <*https://master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts/*> erişim tarihi 2021/11/12.

Url-05. <*https://archello.com/project/museo-soumaya*> erişim tarihi 2022/01/28.

Url-06. <*https://numpy.org/doc/stable/*>
erişim tarihi 2021/11/18.

Url-07. <*https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1*> erişim tarihi 2022/02/02.

Url-08. <*https://algorithmicdesign.github.io/*> erişim tarihi 2022/02/20.

Url-09. <*https://www.nasdaq.com/articles/the-machine-learning-future-is-now%3a-how-ai-is-disrupting-entire-industries*> erişim tarihi 2022-08-14.

Url-10. <*https://devhunteryz.wordpress.com/2018/06/20/geri-yayilimbackpropagation/*> erişim tarihi 2022/05/10.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33– 35.

Woodbury, R. (2010). *Elements of parametric design*. Taylor; Francis.

Zhao, C.-W., Yang, J., & Li, J.-T. (2021). Generation of hospital emergency depart- ment layouts based on generative adversarial networks. *Journal of Building Engineering*, 102539.

Zhou, S. (2018). https://www.coursera.org/learn/build-basic-generative-adversarial-networks-gans/
erişim tarihi 2021/09/18

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image trans- lation using cycle-consistent adversarial networks. *Proceedings of the IEEE international conference on computer vision*, 2223–2232.

EK A. ÇALIŞMAYA YÖNELİK KAYNAK KODLAR

```
from tkinter import *
from PIL import Image, ImageDraw
import random
```

Listing A.1 trainingdata.py 1/3

```
def model(a,b,width,height,c):
    white = (255, 255, 255)
    blue = (0,255,255)
    yellow = (255,255,0)
    red = (139,0,0)
    black = (0,0,0)

    image1 = Image.new("RGB", (width, height), white)
    draw = ImageDraw.Draw(image1)
    image2 = Image.new("RGB", (width, height), white)
    draw2 = ImageDraw.Draw(image2)

    draw.rectangle([0, height, width, 0], black)
    draw2.rectangle([0, height, width, 0], red)
    draw2.rectangle([0, height, width/2+a, 0], yellow)
    draw2.rectangle([a, height, width, b], blue)

    filename = "train\\target\\train_{}.jpg".format(c)
    image1.save(filename)
    filename = "train\\input\\train_{}.jpg".format(c)
    image2.save(filename)
```

Listing A.2 rainingdata.py 2/3

```
i=0
control=[]
for i in range(5):
    traindata=[]
    width = random.randrange(60, 180)
    if width % 2 == 0:
        traindata.append(width)
        height = random.randrange(60, 90)
        traindata.append(height)
        if traindata in control:
```

```
            None
        else:
            control.append(traindata)
            a = random.randrange(int(width/12),
                                 int(width/2-width/9))
            b = random.randrange(int(height/12),
                                 int(height-height/12))
            model(a,b,width,height,i)
```

Listing A.3 trainingdata.py 3/3

```
import numpy as np
import config
import os
from PIL import Image
from torch.utils.data import Dataset, DataLoader
from torchvision.utils import save_image


class MapDataset(Dataset):
    def __init__(self,root_dir):
        self.root_dir = root_dir
        self.list_files = os.listdir(self.root_dir)

    def __len__(self):
        return len(self.list_files)

    def __getitem__(self, index):
        img_file = self.list_files[index]
        img_path = os.path.join(self.root_dir, img_file)
        image = np.array(Image.open(img_path))

        input_image = image[:, :int(image.shape[1]/2), :]
        target_image = image[:, int(image.shape[1]/2):, :]

        augmentations = \
            config.both_transform(
                image=input_image,image0=target_image)
        input_image = augmentations["image"]
        target_image = augmentations["image0"]

        input_image = \
            config.transform_only_input(image=input_image)["image"]
        target_image = \
            config.transform_only_mask(image=target_image)["image"]

        return input_image, target_image
```

```
if __name__ == "__main__":
    dataset = MapDataset("train\\")
    loader = DataLoader(dataset, batch_size=5)
    for x, y in loader:
        print(x.shape)
        save_image(x, "x.png")
        save_image(y, "y.png")
        import sys

        sys.exit()
```

Listing A.4 dataset.py

```
import torch
import torch.nn as nn


class Block(nn.Module):
    def __init__(self, in_channels, out_channels,
                 down=True, act="relu", use_dropout=False):
        super(Block, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, 4, 2, 1,
                      bias=False, padding_mode="reflect")
            if down
            else nn.ConvTranspose2d(in_channels, out_channels,
                                    4, 2, 1, bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU() if act == "relu" else nn.LeakyReLU(0.2),
        )

        self.use_dropout=use_dropout
        self.dropout = nn.Dropout(0.5)
        self.down = down

    def forward(self, x):
        x = self.conv(x)
        return self.dropout(x) if self.use_dropout else x


class Generator(nn.Module):
    def __init__(self, in_channels=3, features=64):
        super().__init__()
        self.initial_down = nn.Sequential(
            nn.Conv2d(in_channels, features, 4, 2, 1,
                      padding_mode="reflect"),
            nn.LeakyReLU(0.2),
        )
        self.down1 = Block(features, features * 2,
```

```
                    down=True, act="leaky",
                    use_dropout=False)
self.down2 = Block(
    features * 2, features * 4,
    down=True, act="leaky", use_dropout=False
)
self.down3 = Block(
    features * 4, features * 8,
    down=True, act="leaky", use_dropout=False
)
self.down4 = Block(
    features * 8, features * 8,
    down=True, act="leaky", use_dropout=False
)
self.down5 = Block(
    features * 8, features * 8,
    down=True, act="leaky", use_dropout=False
)
self.down6 = Block(
    features * 8, features * 8,
    down=True, act="leaky", use_dropout=False
)
self.bottleneck = nn.Sequential(
    nn.Conv2d(features * 8, f
eatures * 8, 4, 2, 1), nn.ReLU()
)

self.up1 = Block(features * 8, features * 8,
                 down=False,
                 act="relu", use_dropout=True)
self.up2 = Block(
    features * 8 * 2, features * 8, down=False,
    act="relu",use_dropout=True
)
self.up3 = Block(
    features * 8 * 2, features * 8, down=False,
    act="relu",use_dropout=True
)
self.up4 = Block(
    features * 8 * 2, features * 8, down=False,
    act="relu",use_dropout=False
)
self.up5 = Block(
    features * 8 * 2, features * 4, down=False,
    act="relu",use_dropout=False
)
self.up6 = Block(
    features * 4 * 2, features * 2, down=False,
```

```
            act="relu", use_dropout=False
        )
        self.up7 = Block(features * 2 * 2, features,
                         down=False,
                         act="relu",use_dropout=False)
        self.final_up = nn.Sequential(
            nn.ConvTranspose2d(features * 2, in_channels,
                               kernel_size=4,
                               stride=2, padding=1),
            nn.Tanh(),
        )

    def forward(self, x):
        d1 = self.initial_down(x)
        d2 = self.down1(d1)
        d3 = self.down2(d2)
        d4 = self.down3(d3)
        d5 = self.down4(d4)
        d6 = self.down5(d5)
        d7 = self.down6(d6)
        bottleneck = self.bottleneck(d7)
        up1 = self.up1(bottleneck)
        up2 = self.up2(torch.cat([up1, d7], 1))
        up3 = self.up3(torch.cat([up2, d6], 1))
        up4 = self.up4(torch.cat([up3, d5], 1))
        up5 = self.up5(torch.cat([up4, d4], 1))
        up6 = self.up6(torch.cat([up5, d3], 1))
        up7 = self.up7(torch.cat([up6, d2], 1))
        return self.final_up(torch.cat([up7, d1], 1))


def test():
    x = torch.randn((1, 3, 256, 256))
    model = Generator(in_channels=3, features=64)
    preds = model(x)
    print(preds.shape)


if __name__ == "__main__":
    test()
```

Listing A.5 generatormodel.py

```
import torch
import torch.nn as nn


class CNNBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride):
```

```
        super(CNNBlock, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(
                in_channels, out_channels, 4, stride, 1,
                bias=False, padding_mode="reflect"
            ),
            nn.BatchNorm2d(out_channels),
            nn.LeakyReLU(0.2),
        )

    def forward(self, x):
        return self.conv(x)


class Discriminator(nn.Module):
    def __init__(self, in_channels=3,
                 features=[64, 128, 256, 512]):
        super().__init__()
        self.initial = nn.Sequential(
            nn.Conv2d(
                in_channels * 2,
                features[0],
                kernel_size=4,
                stride=2,
                padding=1,
                padding_mode="reflect",
                ),
            nn.LeakyReLU(0.2),
        )

        layers = []
        in_channels = features[0]
        for feature in features[1:]:
            layers.append(
                CNNBlock(in_channels, feature,
                         stride=
                         1 if feature == features[-1] else 2),
            )
            in_channels = feature

        layers.append(
            nn.Conv2d(
                in_channels, 1, kernel_size=4,stride=1,
                padding=1, padding_mode="reflect"
            ),
        )

        self.model = nn.Sequential(*layers)
```

```
    def forward(self, x, y):
        x = torch.cat([x, y], dim=1)
        x = self.initial(x)
        x = self.model(x)
        return x


def test():
    x= torch.randn((1, 3, 256, 256))
    y= torch.randn((1, 3, 256, 256))
    model = Discriminator()
    preds = model(x, y)
    print(preds.shape)


if __name__ == "__main__":
    test()
```

Listing A.6 discriminatormodel.py

```
import torch
import config
from torchvision.utils import save_image

def save_some_examples(gen, val_loader,epoch, folder):
    x, y = next(iter(val_loader))
    x, y = x.to(config.DEVICE), y.to(config.DEVICE)
    gen.eval()
    with torch.no_grad():
        y_fake = gen(x)
        y_fake = y_fake * 0.5 + 0.5
        save_image(y_fake,
                   folder + f"/y_gen_{epoch}.png")
        save_image(x * 0.5 + 0.5,
                   folder + f"/input_{epoch}.png")
        if epoch == 1:
            save_image(y * 0.5 + 0.5,
                       folder + f"/label_{epoch}.png")
    gen.train()


def save_checkpoint(model, optimizer,
                    filename="my_checkpoint.pth.tar"):
    print("=>heckpoint")checkpoin
    t = {
        "state_dict": model.state_dict(),
        "optimizer": optimizer.state_dict(),
    }
```

```
    torch.save(checkpoint, filename)


def load_checkpoint(checkpoint_file, model, optimizer, lr):
    print("=>heckpoint")
    checkpoint = torch.load(checkpoint_file,
                            map_location=config.DEVICE)
    model.load_state_dict(checkpoint["state_dict"])
    optimizer.load_state_dict(checkpoint["optimizer"])

    for param_group in optimizer.param_groups:
        param_group["lr"] = lr
```

Listing A.7 utils.py

```
import torch
import albumentations as A
from albumentations.pytorch import ToTensorV2

DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
TRAIN_DIR = "train\\"
VAL_DIR = "val\\"
LEARNING_RATE = 2e-4
BATCH_SIZE = 16
NUM_WORKERS = 2
IMAGE_SIZE = 256
CHANNELS_IMG = 3
L1_LAMBDA = 500
LAMBDA_GP = 10
NUM_EPOCHS = 1000
LOAD_MODEL = False
SAVE_MODEL = True
CHECKPOINT_DISC = "disc.pth.tar"
CHECKPOINT_GEN = "gen.pth.tar"

both_transform = A.Compose(
    [A.Resize(width=256, height=256),],
    additional_targets={"image0": "image"},
)

transform_only_input = A.Compose(
    [
        A.HorizontalFlip(p=0.5),
        A.ColorJitter(p=0.2),
        A.Normalize(mean=[0.5, 0.5, 0.5],
                    std=[0.5, 0.5, 0.5],
                    max_pixel_value=255.0,),
        ToTensorV2(),
    ]
```

```
)

transform_only_mask = A.Compose(
    [
        A.Normalize(mean=[0.5, 0.5, 0.5],
                    std=[0.5, 0.5, 0.5],
                    max_pixel_value=255.0,),
        ToTensorV2(),
    ]
)
```

Listing A.8 config.py

```
import torch
from utils import *
import torch.nn as nn
import torch.optim as optim
import config
from dataset import MapDataset
from generator_model import Generator
from discriminator_model import Discriminator
from torch.utils.data import DataLoader
from tqdm import tqdm
from torchvision.utils import save_image


def train_fn(
        disc, gen, loader, opt_disc, opt_gen,
        l1_loss, bce, g_scaler, d_scaler,
):
    loop = tqdm(loader, leave=True)

    for idx, (x, y) in enumerate(loop):
        x = x.to(config.DEVICE)
        y = y.to(config.DEVICE)

        # Train Discriminator
        with torch.cuda.amp.autocast():
            y_fake = gen(x)
            D_real = disc(x, y)
            D_real_loss = bce(D_real,
                              torch.ones_like(D_real))
            D_fake = disc(x, y_fake.detach())
            D_fake_loss = bce(D_fake,
                              torch.zeros_like(D_fake))
            D_loss = (D_real_loss + D_fake_loss) / 2

        disc.zero_grad()
        d_scaler.scale(D_loss).backward()
```

```
        d_scaler.step(opt_disc)
        d_scaler.update()

        # Train generator
        with torch.cuda.amp.autocast():
            D_fake = disc(x, y_fake)
            G_fake_loss = bce(D_fake,
                              torch.ones_like(D_fake))
            L1 = l1_loss(y_fake, y) * config.L1_LAMBDA
            G_loss = G_fake_loss + L1

        opt_gen.zero_grad()
        g_scaler.scale(G_loss).backward()
        g_scaler.step(opt_gen)
        g_scaler.update()

        if idx % 10 == 0:
            loop.set_postfix(
                D_real=torch.sigmoid(D_real).mean().item(),
                D_fake=torch.sigmoid(D_fake).mean().item(),
            )


def main():
    disc = Discriminator(in_channels=3).to(config.DEVICE)
    gen = Generator(in_channels=3,
                    features=64).to(config.DEVICE)
    opt_disc = optim.Adam(disc.parameters(),
                          lr=config.LEARNING_RATE,
                          betas=(0.5,0.999),)
    opt_gen = optim.Adam(gen.parameters(),
                         lr=config.LEARNING_RATE,
                         betas=(0.5,0.999))
    BCE = nn.BCEWithLogitsLoss()
    L1_LOSS = nn.L1Loss()

    if config.LOAD_MODEL:
        load_checkpoint(
            config.CHECKPOINT_GEN, gen,
            opt_gen, config.LEARNING_RATE,
        )
        load_checkpoint(
            config.CHECKPOINT_DISC, disc,
            opt_disc, config.LEARNING_RATE,
        )

    train_dataset = MapDataset(root_dir=config.TRAIN_DIR)
    train_loader = DataLoader(
```

```
        train_dataset,
        batch_size=config.BATCH_SIZE,
        shuffle=True,
        num_workers=config.NUM_WORKERS,
    )
    g_scaler = torch.cuda.amp.GradScaler()
    d_scaler = torch.cuda.amp.GradScaler()
    val_dataset = MapDataset(root_dir=config.VAL_DIR)
    val_loader = DataLoader(val_dataset,
                            batch_size=1, shuffle=False)

    for epoch in range(config.NUM_EPOCHS):
        train_fn(
            disc, gen, train_loader, opt_disc, opt_gen,
            L1_LOSS, BCE, g_scaler, d_scaler,
        )

        if config.SAVE_MODEL and epoch % 5 == 0:
            save_checkpoint(gen, opt_gen,
                            filename=config.CHECKPOINT_GEN)
            save_checkpoint(disc, opt_disc,
                            filename=config.CHECKPOINT_DISC)

        save_some_examples(gen, val_loader, epoch,
                           folder="evaluation")


if __name__ == "__main__":
    main()
```

Listing A.9 train.py

```
import torch
import torch.nn as nn

linear = nn.Linear(10, 2        )
example_input = torch.randn(3, 10)
example_output = linear(example_input)

print(example_output)
```

Listing A.10 main.py

```
import setuptools

with open("README.md", "r") as fh:
    long_description = fh.read()

setuptools.setup(
    name="image-similarity-measures",
```

```
    version="0.3.5",
    author="UP42",
    author_email="",
    description="",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://github.com/up42/image-similarity-measures",
    packages=setuptools.find_packages(),
    license="MIT",
    classifiers=[
        "Programming::Python::3",
        "License::OSI::MITicense","Operating::OSnd
        ependent",
        "Intended::Science/Research",
        "Developmenttatus::5roduction/Stable",
    ],
    extras_require={
        "rasterio": ["rasterio"],
        "speedups": ["pyfftw"],
    },
    install_requires=["numpy", "scikit-image",
                      "opencv-python", "phasepack"],
    python_requires=">=3.6,3.10",
    entry_points={
        "console_scripts": [
            "image-similarity-measures="
            "image_similarity_measures.evaluate:main"
        ],
    },
)
```

Listing A.11 setup.py