T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

İKİ BOYUTLU RESİM SERİLERİNDEN MAKİNE ÖĞRENMESİ KULLANARAK ÜÇ BOYUTLU MODEL OLUŞTURMA

Batuhan AŞIROĞLU

DANIŞMAN
Dr. Öğr. Üyesi Mustafa DAĞTEKİN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Aralık, 2022

TEZ KABUL VE ONAYI

Batuhan AŞIROĞLU tarafından, **Dr. Öğr. Üyesi Mustafa DAĞTEKİN** danışmanlığında hazırlanan **"İKİ BOYUTLU RESİM SERİLERİNDEN MAKİNE ÖĞRENMESİ KULLANILARAK ÜÇ BOYUTLU MODEL OLUŞTURMA"** başlıklı bu çalışma, jürimiz tarafından **30/12/2022** tarihinde yapılan sınav sonucunda **oy birliği** ile başarılı bulunarak **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Tez Jürisi

		İmza	**Sonuç**
DANIŞMAN	Dr. Öğr. Üyesi Mustafa DAĞTEKİN İstanbul Üniversitesi-Cerrahpaşa Bilgisayar Donanımı ve Gömülü Sistemler Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Doç. Dr. Sibel SENAN İstanbul Üniversitesi-Cerrahpaşa Üniversitesi Bilgisayar Bilimleri Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Dr. Öğr. Üyesi Ertuğrul SAATÇİ İstanbul Kültür Üniversitesi Elektrik-Elektronik Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret

Aileme ithaf ediyorum…

BÜTÇE DESTEKLERİ

İKİ BOYUTLU RESİM SERİLERİNDEN MAKİNE ÖĞRENMESİ KULLANILARAK ÜÇ BOYUTLU MODEL OLUŞTURMA

Bu tez çalışması için herhangi bir kurumdan bütçe desteği alınmamıştır.

TEŞEKKÜR

Bana kendimi geliştirmemde, bilgi birikimi, deneyim kazanmamda, yetişmemde ve gelişmemde desteği olan aileme, başöğretmen Mustafa Kemal ATATÜRK başta olmak üzere bütün hocalarıma ve arkadaşlarıma teşekkür ederim.

Aralık 2022 Batuhan AŞIROĞLU

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Kısaltmalar	Açıklama
YSA	: Yapay Sinir Ağları
ESA	: Evrişimli Sinir Ağları
CNN	: Convolutional Neural Networks
MSE	: Mean Squared Error
RMSE	: Root Mean Squared Error
MAE	: Mean Absolute Error
BN	: Batch Normalization
RGB	: Red Green Blue
CMYK	: Cyan Magenta Yellow Key
HSV	: Hue Saturation Value
ResNet	: Residual Network
Relu	: Rectified Linear Unit
CE	: Cross Entropy
CMCE	: Categorical Mean Cross Entropy
Conv	: Convolution
TConv	: Transposed Convolution
IoU	: Intersection Over Union

ÖZET

YÜKSEK LİSANS TEZİ

İKİ BOYUTLU RESİM SERİLERİNDEN MAKİNE ÖĞRENMESİ KULLANILARAK ÜÇ BOYUTLU MODEL OLUŞTURMA

Batuhan AŞIROĞLU

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman : Dr. Öğr. Üyesi Mustafa DAĞTEKİN

Günümüzde dijital ikiz, oyun geliştirme, Metaverse gibi alanlarda kullanılması amacıyla gerçek dünyadaki nesneleri sanal dünyaya hızlı bir şekilde aktarılması ihtiyacı doğmaktadır. Bu problem için hem tek açıdan çekilen görüntüler kullanarak hem de birden çok açıdan çekilen görüntüler kullanarak 3D modeller üretebilen derin öğrenme tabanlı çalışmalar mevcuttur. Bu tezde yeni bir derin öğrenme ağı ve yeni hata fonksiyonu geliştirilerek, 4 farklı açıdan çekilen 128x128 boyutlu görüntülerden 7 kategoride (araba, uçak, kanepe, sandalye, masa, lamba ve silah) ve kategorik olarak denetimsiz, ortalama 0.5283 IoU skor ile voxel gösterimli 3D model üretimi yapıldığı anlatılmaktadır.

Aralık 2022, 66 sayfa.

Anahtar kelimeler: 3D Yeniden Oluşturma, Derin Öğrenme, Makine Öğrenmesi, Görüntü İşleme, Bilgisayar Görü, Voxel

ABSTRACT

M.Sc. THESIS

3D MODEL GENERATION FROM 2D IMAGE SEQUENCES USING MACHINE LEARNING

Batuhan AŞIROĞLU

İstanbul University-Cerrahpaşa

Institute of Graduate Studies

Department of Computer Engineering

Program of Computer Engineering

Supervisor : Assist. Prof. Dr. Mustafa DAĞTEKİN

Nowadays, there is a need to quickly transfer real word objects to the virtual world in order to use them in areas such as digital twin, game development and Metaverse. To accomplish this, there are deep learning based studies that can generate 3D models while using both images taken at a single angle and at multiple angles. This thesis explains that voxel representational 3D models were produced in 7 categories such as car, airplane, sofa, chair, table, lamb and gun from 128x128 sized images taken from 4 different angles as categorically unsupervised with 0.5283 average IoU score by developing novel deep learning network and novel loss function.

December 2022, 66 pages.

Keywords: 3D Reconstruction, Deep Learning, Machine Learning, Image Processing, Computer Vision, Voxel

1. GİRİŞ

Günümüzde dijital ikiz, oyun geliştirme, simülasyon oluşturma, Metaverse vb. gibi alanlarda 3 boyutlu modeller kullanılmaktadır [1, 5]. Ayrıca 3 boyutlu modellerden, mühendislik sektöründe, ürünlerin üretimi sırasında, ürünlerin görünümünün, işlevselliğinin ve özelliklerinin tasarlanmasında [1], endüstriyel otomasyon ve robotik sistemlerin tasarımı ve kurulmasında [2, 3], sağlık sektöründe, tıbbi görüntüleme yöntemleriyle elde edilen görüntülerin işlenmesi ile doktorların daha iyi çıkarımlar yapmasında ve cerrahi işlemlerin planlamasında [4], haritalama sektöründe, daha gerçekçi ve doğru haritaların çıkarılması ve gelecek planlarının yapılmasında [5] da faydalanılmaktadır. Bütün bunlar birlikte ele alınınca gerçek dünyadaki nesneleri dijital ortama taşıma gerekliliği ortaya çıkmıştır. Bu ihtiyacı gidermek amacı ile gerçek dünyadaki nesnelerin bilgisayar ortamında 3 boyutlu modellemeleri üretilmektedir.

3 boyutlu modellemelerin üretiminin bilgisayar ortamında elle yapılması işlemi oldukça uzun sürmektedir. Bu sebeple bilgisayar görü algoritmaları ile gerçek dünyadaki nesnelerin 2 boyutlu fotoğrafı veya fotoğrafları ile 3 boyutlu modellemesi üretilmesi problemi ortaya çıkmıştır. 2 boyutlu fotoğraf kullanılarak 3 boyutlu modelleme üretilme problemi günümüzde büyük bir zorluk olmaktadır. Bununla beraber derin öğrenme metotları kullanılarak yapılan nesne tanıma [6, 7] ve nesne tespiti [8, 9] çalışmaları da her geçen gün önem kazanmaktadır. Araştırmacılar bilgisayar görü algoritmaları ve derin öğrenme algoritmaları ile yapılan son çalışmalarda [10-12, 14] bu problemi çözmek adına oldukça önemli sonuçlar elde etmişlerdir.

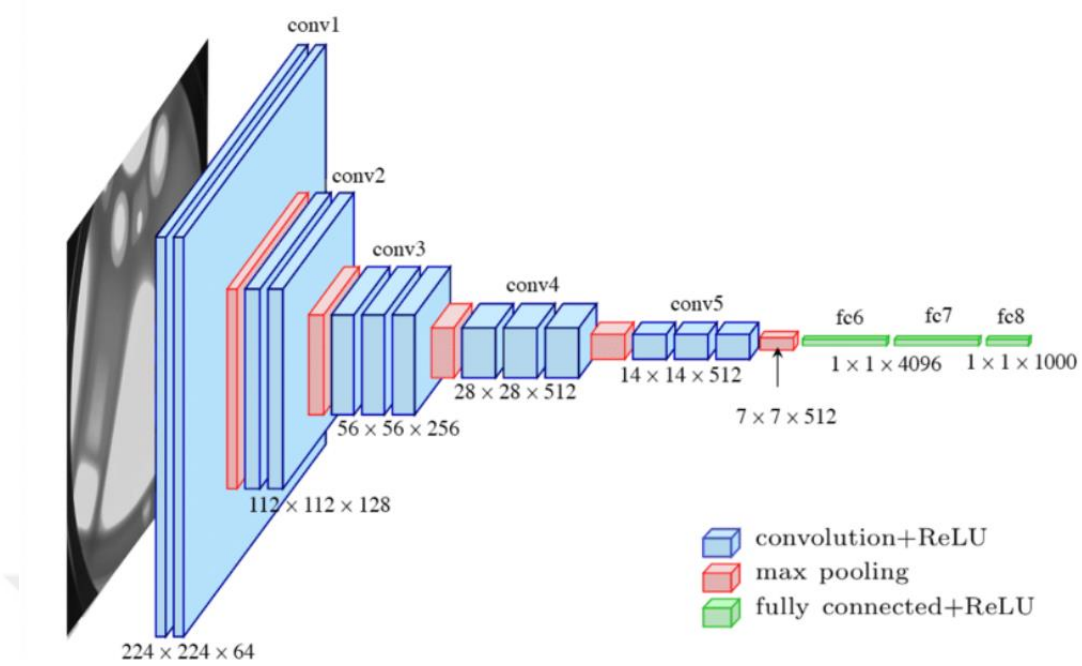
Derin öğrenme ile 3 boyutlu yeniden oluşturma problemi için hem girdi sayısı olarak tek görüntü kullanarak 3 boyutlu modeller oluşturan çalışmalar [10, 15-17] hem de girdi sayısı olarak birden çok görüntü kullanarak 3 boyutlu modeller oluşturan çalışmalar [16, 17] araştırmacılar tarafından yapılmıştır. Ancak bu çalışmalar kategorik olarak denetimli olacak şekilde yapılmıştır. Bunun yanında Gwak ve diğ. önerdiği McRecon [16] ve Yan ve diğ. önerdiği PTN [17] yöntemleri ShapeNet veri kümesini kullanarak kategorik olarak denetimli bir şekilde Voxel tabanlı 3 boyutlu modeller üretmeyi başarmıştır.

Bu tezde 4 farklı açıdan çekilmiş 128×128 boyutlu 2 boyutlu görüntüler kullanılarak derin öğrenme ile Voxel gösterimli 3 boyutlu model oluşturulduğu anlatılmaktadır. Kavramsal çerçeve bölümünde literatürdeki çalışmalar, Malzeme ve Yöntem bölümünde bu tez yapılırken faydalanılan ve geliştirilen metotlar, Bulgular bölümünde tez çalışması sonunda elde edilen bulgular, Tartışma bölümünde tez çalışması sonucunda elde edilen bulguların literatürdeki diğer çalışmalar karşılaştırılması, Sonuç ve Öneriler bölümünde tez çalışmasının sonuçları, eksik ve geliştirilmesi gereken yönleri ele alınmıştır.

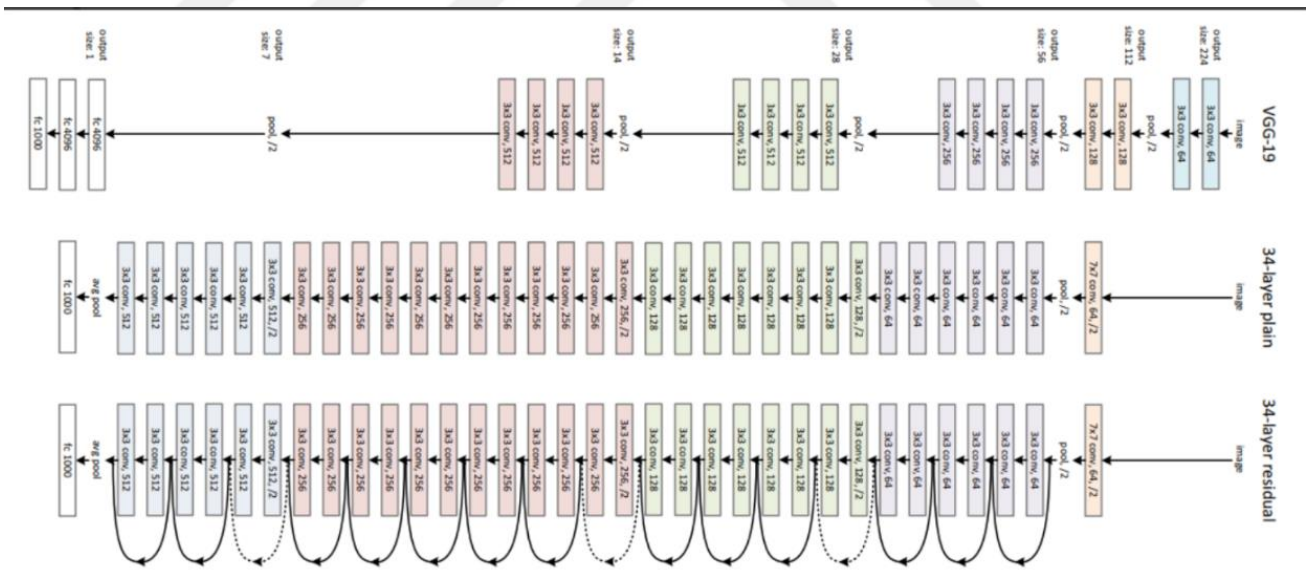
2. KAVRAMSAL ÇERÇEVE

Günümüzde artan işlem gücü ile yapay zeka algoritmaları artık daha gerçeklenebilir hale gelmiştir. Yapay zeka teknolojileri özellikle makine öğrenmesi ve makine öğrenmesinin alt kümesi olan derin öğrenme metotları (Şekil 2.1), her geçen gün bir dilden başka bir dile çeviri, nesne tanıma, nesne tespiti gibi karmaşık örüntüler içeren görevlerin üstesinden daha başarılı bir şekilde gelmektedir. Derin öğrenme algoritmalarının gelişmesi ve karmaşık örüntülerden anlam çıkarıp belirli bir amaca yönelik işlemler yapabilmesi, büyük verinin oluşumu ile doğru orantılıdır. Bunun sebebi derin öğrenme algoritmaları yapıları gereği binlerce, milyonlarca ve hatta bazı problemler için milyarlarca parametre içermektedirler. Derin öğrenme algoritmalarının bu fazla parametreleri ayarlama/bulma (tune) işlemi ancak büyük veri ile başarı ile yapılabilmektedir. VGG [18], Resnet [20] (Şekil 2.2 ve Şekil 2.3) gibi nesne tanıma (object recognition) problemi için geliştirilen modern derin öğrenme tabanlı evrişimli sinir ağları, büyük veri kullanarak eğitilmiş derin öğrenme algoritmalarından bazılarıdır. VGG, Resnet gibi daha önceden büyük veri ile eğitilen derin öğrenme ağları, eğitim amaçlarının dışında nesne bulma (object detection), segmentasyon gibi farklı problemleri de modellemek için öğrenim aktarımı (transfer learning) [21] metodu ile, eğitilecek yeni derin öğrenme mimarilerin öznitelik çıkartma (feature extraction) katmanı olarak da kullanılması yönünden oldukça önem taşımaktadır. Öğrenim aktarımı yöntemi, bölüm 3.6'da detaylı olarak ele alınmıştır.

Şekil 2.1: Derin Öğrenme, Makine Öğrenmesi ve Yapay Zeka İlişkisi



Şekil 2.2: VGG Mimarisi [18]



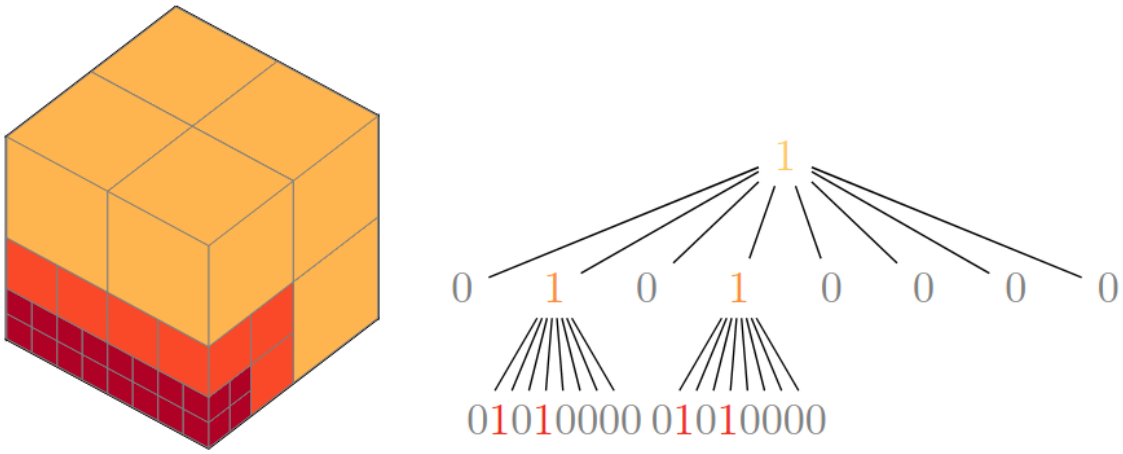
Şekil 2.3: Resnet Mimarisi [22]

2.1. 3D YENİDEN OLUŞTURMA (3D RECONSTRUCTION)

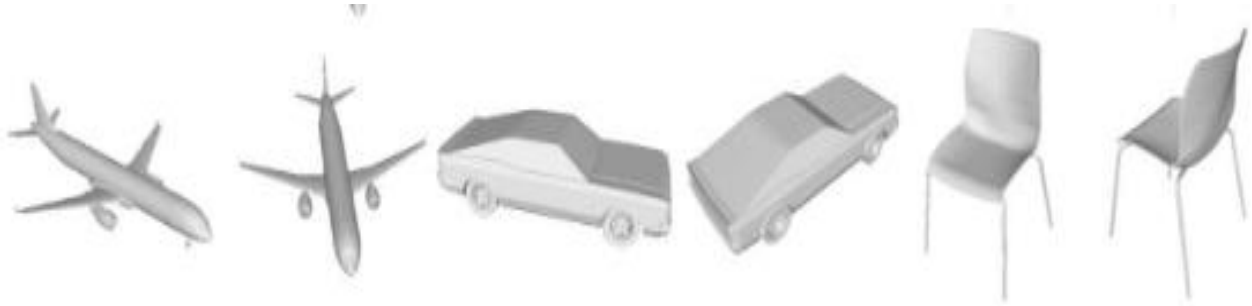
3D yeniden oluşturma, bir nesnenin veya sahnenin (scene) 2D görüntülerden veya ölçümlerden oluşan bir dizi kullanılarak 3D modelinin oluşturulması işlemidir.

Günümüzde araştırmacılar tarafından, 2 boyutlu resimlerden 3 boyutlu modelleme oluşturmak için hem bilgisayar görü hem de derin öğrenme tabanlı yaklaşımlar [6, 8-11] kullanılmıştır.

Yapılan çalışmalarda derin öğrenme tabanlı yaklaşımlarda 3 boyutlu model gösterimi için genel olarak nokta bulutu (point cloud) gösterimi (Şekil 2.7) ve voxel gösterimi (Şekil 2.6) olmak üzere iki gösterim türü tercih edilmiştir. Voxel tabanlı 3 boyutlu model gösterimi bölüm 2.1.1'de, nokta bulutu tabanlı 3 boyutlu gösterim bölüm 2.1.2'de detaylıca ele alınmıştır. Bu iki gösterim dışında üç boyutlu modelin koordinatlarının 8'li ağaçlar ile gösterildiği octree (Şekil 2.4) ve üç boyutlu modelin poligonlar ile oluşturulduğu mesh gösterim (Şekil 2.5) de kullanılmıştır ancak nokta bulutu ve voxel gösterimleri daha fazla kullanılmaktadır. Ayrıca voxel ve nokta bulutu gösterimi için mesh ve octree gösterimine göre daha fazla veri kümesi mevcuttur. Bununla beraber Yuniarti ve diğ. [5] ağaç tabanlı octree gösterimi diğer gösterimlere göre daha verimli bir gösterim olduğunu, octree gösteriminde daha az veri kullanılarak ile daha fazla ayrıntı gösterilebildiğini belirtmiştir. Bazı araştırmacılar yaptıkları çalışmalarda, 3D reconstruction problemi için birden fazla açıdan çekilmiş/üretilmiş resimler kullanarak 3D model oluşturmayı [10, 11] tercih etmiştir. Bazı araştırmacılar ise aynı problem için tek bir açıdan çekilmiş/oluşturulmuş görüntü üzerinden 3D model oluşturmak amaçlı [6, 7, 9] algoritmalar geliştirmiştir



Şekil 2.4: Octree Gösterim [24]

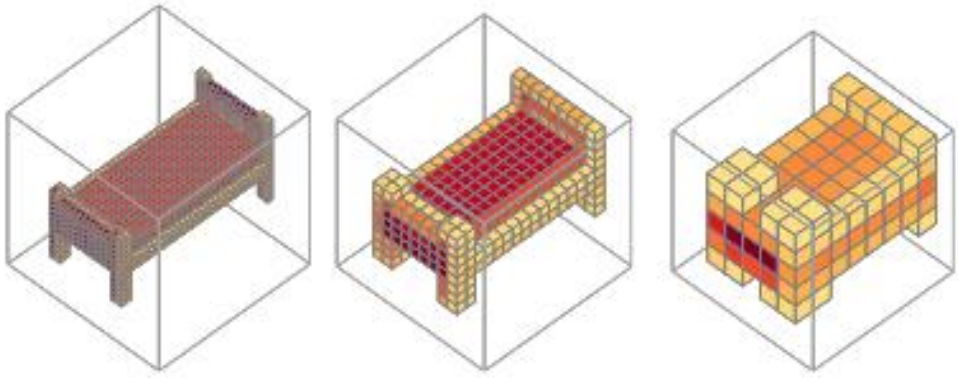


Şekil 2.5: Mesh Gösterim [25]

2.1.1. Voxel Tabanlı 3D Model Gösterimi

Voxel Tabanlı 3 boyutlu model gösterimde (Şekil 2.6) modellerin üç boyutlu uzayda X, Y, Z koordinatlarındaki pixel değerleri ile oluşturulan gösterim şeklidir. Voxeller 3 boyutlu grafiklerde küpler ile temsil edilmektedir. Voxel tabanlı 3 boyutlu model gösterimi diğer gösterim şekillerine göre oluşturulan 3 boyutlu modellerden daha fazla veri ile temsil edilir [23]. Bunun yanında Voxeller, 3 boyutlu grafiklerdeki noktaları temsil eder ve her voxel birbirinden bağımsız olarak değiştirilebilir ve yönetilebilir. Bu da Voxellerin 3 boyutlu grafikler için esnek ve çok yönlü bir yol olmasını sağlamaktadır.

Bu gösterim şekli için Chang ve diğ. [11] oluşturduğu ShapeNet ve ShapeNetCore veri kümeleri mevcuttur. ShapeNet veri kümesi 14 kategoride toplam 43.784 adet veri içermekte, ShapeNet veri kümesinin alt kümesi olan ShapeNetCore veri kümesinde ise 55 kategoride 51,300 adet 3D model içermektedir. ShapeNet veri kümesinde her 3D voxel modeli için 24 farklı açıdan render alınmış 137×137 boyutlu RGB görüntüler bulunmaktadır. Bu çalışmada önerilen derin öğrenme modelinin eğitiminde ShapeNet veri kümesi kullanılmıştır.



Şekil 2.6: Voxel Gösterim [24]

2.1.2. Nokta Bulutu Tabanlı 3D Model Gösterimi

Nokta bulutu (point cloud) gösterimi (Şekil 2.7) 3D gösterimlerin en popülerlerinden biridir [23]. Nokta bulutu gösterimi, 3 boyutlu uzayda nesnelerin yüzeylerini noktalar bütünüyle temsil edildiği gösterim şeklidir. Nokta bulutu, uzaydaki veri noktalarının ayrık (discrete) bir kümesidir ve her nokta 3 boyutlu koordinat sisteminde (X, Y, Z), temsil ettiği objenin karşılığı olan koordinatı göstermektedir.

Nokta bulutu gösterimi için Behley ve diğ. [27] tarafından oluşturulan Kitti veri kümesi mevcuttur. Kitti veri kümesinde 28 kategoride toplam 43.000 adet veri mevcuttur.

Şekil 2.7: Nokta Bulutu Gösteri [28]

3. YÖNTEM

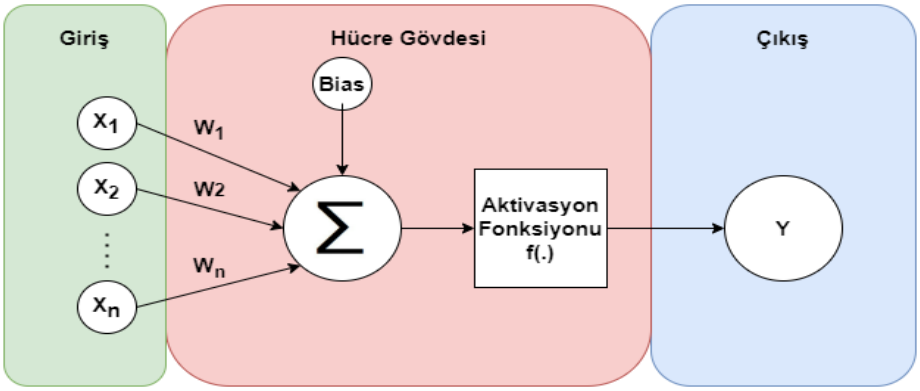
Bu bölümde bu tez çalışması yapılırken faydalanılan ve geliştirilen metotlar anlatılmaktadır.

3.1. YAPAY SİNİR AĞLARI

Yapay sinir ağları (YSA), günümüzde gelişmiş yapay zeka yöntemlerinden biridir ve lineer olmayan karmaşık problemleri çözmek için birçok çalışmada kullanılmaktadır. YSA'lar, insan beynindeki nöronların işleyiş süreçlerinin Denklem 3.1'deki gibi f aktivasyon fonksiyonunda b bias değeri, n adet x girdisi ve her x girdisine karşılık gelen w ağırlıkları ile matematiksel olarak modellenmesi sonucunda oluşturulan ve karmaşık problemlere çözüm üretmeye çalışan sistemlerdir [13].

$$y = f\left(b + \sum_{i=0}^{n} x_i * w_i\right) \tag{3.1}$$

YSA'lardaki bir nöron, giriş, hücre gövdesi (cell body) ve çıkış olmak üzere üç bölümden oluşur (Şekil 3.1). Giriş katmanı tarafından girdi değerleri nörona ulaştırılmaktadır. Nörona ulaşan girdi değerleri, hücre gövdesinde kendi ağırlıkları ile çarpılmakta ve bütün girdiler toplanmaktadır. Bu işlemden sonra hücre gövdesinde üretilen sonuç aktivasyon fonksiyonundan geçirilerek nihai sonuç üretilmektedir. Son olarak çıkış katmanında, hücre gövdesinde üretilen sonuç bir sonraki nörona veya nöronlara iletilmektedir.



Şekil 3.1: Yapay Sinir Hücresi (Perceptron)

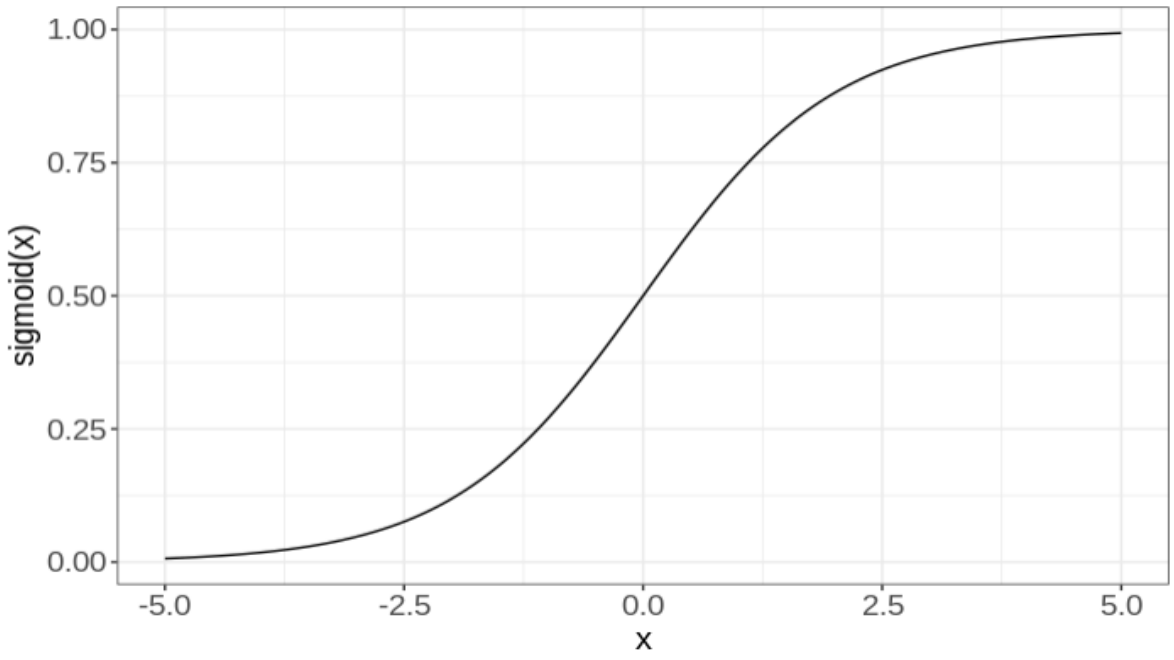
3.1.1. Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları yapay sinir ağlarında oldukça büyük öneme sahiptir. Aktivasyon fonksiyonlarının yapay sinir ağlarına kazandırdığı en büyük avantaj yapılan hesaplamalardaki lineerliği kırmasıdır. Bu yönüyle lineer makine öğrenmesi yöntemlerinden ayrışmaktadır. Ayrıca aktivasyon fonksiyonları yapay sinir ağının çıkış kümesini belli bir aralığa indirgemektedir. Bu açıdan yapay sinir ağlarında kullanılan aktivasyon fonksiyonu problemin türüne, çıkış kümesinin aralığa göre değişmektedir. Aktivasyon fonksiyonlarının en popülerlerinden olan Sigmoid aktivasyon fonksiyonu bölüm 3.1.1.1'de, Tanh aktivasyon fonksiyonu bölüm 3.1.1.2'de, Relu aktivasyon fonksiyonu bölüm 3.1.1.3'te, Leaky Relu aktivasyon fonksiyonu bölüm 3.1.1.4'te ve Softmax aktivasyon fonksiyonu bölüm 3.1.1.5'te ele alınmıştır.

3.1.1.1. Sigmoid

Sigmoid aktivasyon fonksiyonu (Denklem 3.2) 0 ile 1 arasında değer vermektedir (Şekil 3.2). Genellikle sınıflandırma problemlerinde kullanılmaktadır.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

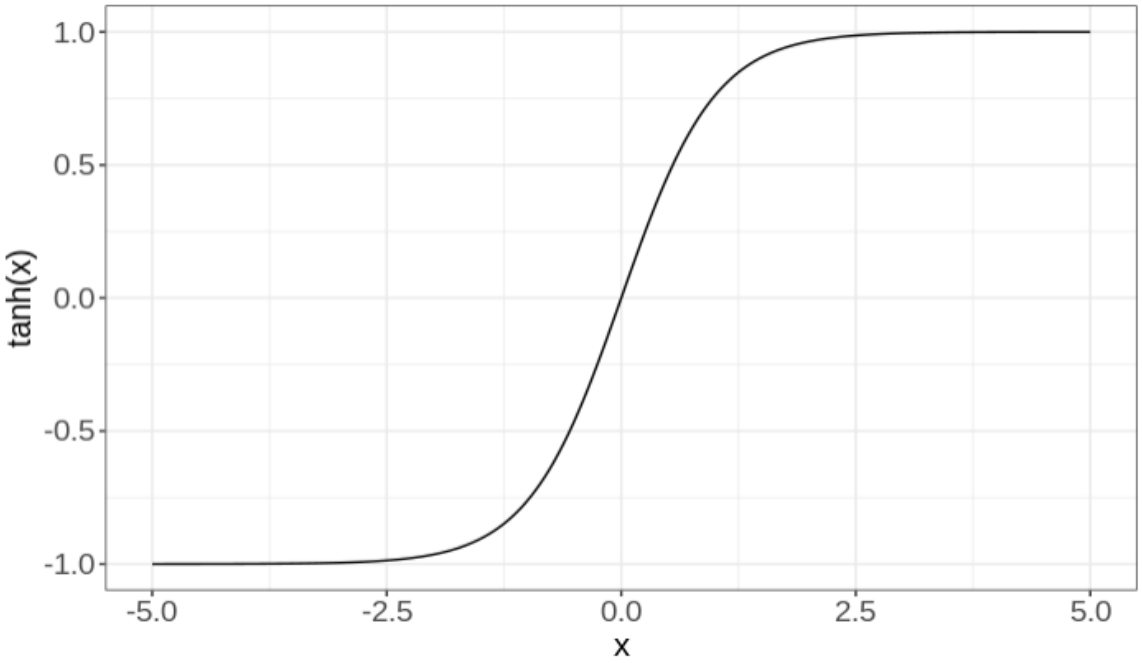


Şekil 3.2: Sigmoid Aktivasyon Fonksiyonu

3.1.1.2. Tanh

Tanh aktivasyon fonksiyonu (Denklem 3.3) -1 ile 1 arasında değer vermektedir (Şekil 3.3). Tanh aktivasyon fonksiyonu birçok problemde tercih edilmektedir.

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.3}$$



Şekil 3.3: Tanh Aktivasyon Fonksiyonu

3.1.1.3. Relu

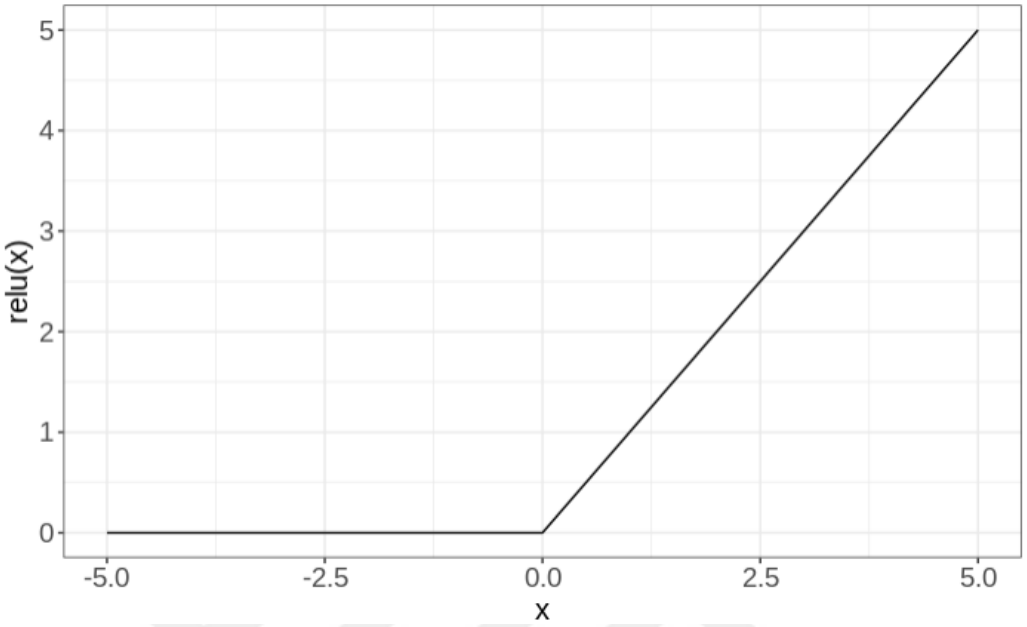
Relu aktivasyon fonksiyonu (Denklem 3.4) 0'dan küçük x değerleri için 0, 0'dan büyük ya da 0'a eşit x değerleri için x değerini üretir (Şekil 3.4). Sürekli verilerin tahminlenmesi problemlerinde veya derin öğrenme algoritmalarında gizli (hidden) katmanlarda oldukça sık kullanılmaktadır.

$$Relu(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{3.4}$$

3.1.1.4. Leaky Relu

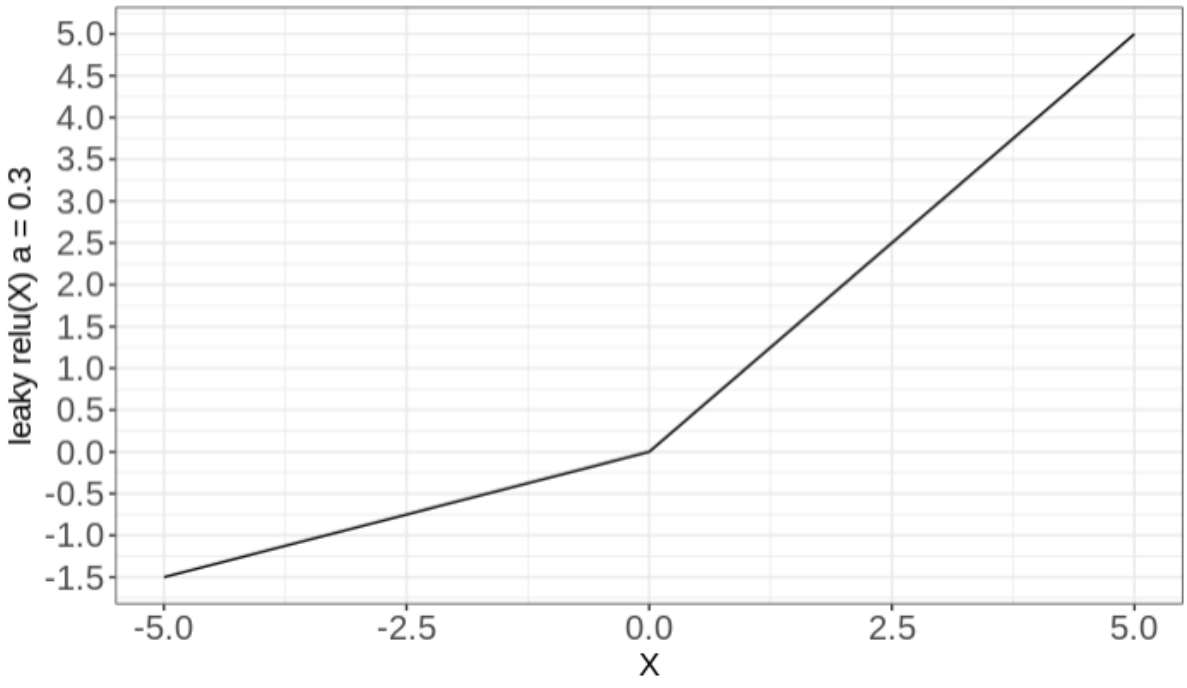
Leaky Relu aktivasyon fonksiyonu (Denklem 3.5) Relu aktivasyon fonksiyonu gibidir ancak 0'dan küçük x değerleri için $0 < a < 1$ olmak üzere $a \cdot x$ değerini üretmektedir. Böylece Leaky Relu aktivasyon fonksiyonu, Relu aktivasyon fonksiyondan farklı olarak 0'dan küçük x değerleri içinde 0'dan farklı bir çıktı üretmektedir (Şekil 3.5). Genelde sürekli verilerin

tahminlenmesi problemlerinde veya derin öğrenme algoritmalarında gizli (hidden) katmanlarda kullanılmaktadır.



Şekil 3.4: Relu Aktivasyon Fonksiyonu

$$LeakyRelu(x) = f(x) = \begin{cases} a \cdot x, & x < 0 \\ x, & x \geq 0 \end{cases}, \quad 0 < a < 1 \tag{3.5}$$



Şekil 3.5: Leaky Relu Aktivasyon Fonksiyonu

3.1.1.5. Softmax

Softmax aktivasyon fonksiyonu (Denklem 3.6), K tane sınıflı sınıflandırma problemlerinde kullanılan ve her sınıfın olasılığını 0 - 1 aralığına indirgeyen bir aktivasyon fonksiyonudur.

$$Softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad , i = 1, \dots, K \tag{3.6}$$

3.1.2. Kayıp - Hata Fonksiyonları

Bir yapay sinir ağı algoritmasında, eğitilen ağın ürettiği sonucun olması gereken değer ve/veya değerlerden ne kadar uzakta olduğunu hesaplamaya yarayan fonksiyonlara kayıp (loss) - hata (error) fonksiyonu adı verilmektedir. Bu fonksiyonlar hem eğitim esnasında hem de eğitim esnasından sonra geliştirilen yapay sinir ağı modelinin ne kadar başarılı olduğu hakkında bilgi vermektedir. Bunun yanında hata fonksiyonları eğitim esnasında istenilen sonuçtan ne kadar uzakta olduğunu belirtip, yapay sinir ağında eğitim için kullanılan optimizasyon algoritmasına parametreleri optimize etmesi için hata değerini geri besler. Bu sebepten dolayı hata fonksiyonun seçilmesi veya oluşturulması problemin çözülebilmesi için oldukça önemlidir. Kayıp - hata fonksiyonlarının en popülerlerinden Ortalama Kare Hata fonksiyonu bölüm 3.1.2.1'de, Kök Ortalama Kare Hata fonksiyonu bölüm 3.1.2.2'de, Ortalama Mutlak Hata fonksiyonu bölüm 3.1.2.3'de ve Çapraz Entropi hata fonksiyonu bölüm 3.1.2.4'de detaylandırılmıştır.

3.1.2.1. Ortalama Kare Hata (Mean Squared Error - MSE)

Ortalama kare hata fonksiyonunda (Denklem 3.7), istenilen n tane çıktı değerleri (y) ile n tane tahmin (predict) edilen değerler ($\hat{y}$) birbirinden çıkarılmakta ve karesi alınmaktadır. Elde edilen sonuç ise toplam veri sayına bölünmektedir. Ortalama kare hata fonksiyonu birçok problem için kullanılabilmektedir ama daha çok hava sıcaklığı tahmini, ev fiyatı tahmini gibi sürekli (continuous) değerlerin tahminlenmesi problemlerinde tercih edilmektedir.

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.7}$$

3.1.2.2. Kök Ortalama Kare Hata (Root Mean Squared Error - RMSE)

Kök ortalama kare hata fonksiyonunda (Denklem 3.8), istenilen çıktı değeri ile tahmin edilen değerler birbirinden çıkarılmakta ve karesi alınmaktadır. Elde edilen sonuç ise toplam veri sayına bölünmekte ve son olarak karekökü alınmaktadır. Kök ortalama kare hata fonksiyonu da

ortalama kare hata fonksiyonu gibi sürekli değerlerin tahminlenmesi problemlerinde tercih edilmektedir.

$$RMSE(y,\hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.8}$$

3.1.2.3. Ortalama Mutlak Hata (Mean Absolute Error - MAE)

Ortalama mutlak hata fonksiyonunda (Denklem 3.9), n tane istenilen çıktı değeri (y) ile n tane tahmin edilen değerler ($\hat{y}$) birbirinden çıkarılmakta ancak kök ortalama kare hata ve ortalama kare hata fonksiyonlarından farklı olarak elde edilen değerin karesi alınması yerine mutlak değeri alınmaktadır ve en son yine toplam veri sayısına bölünmektedir. Bu fonksiyon da kök ortalama kare hata ve ortalama kare hata fonksiyonları gibi sürekli değerlerin tahminlenmesi problemlerinde sıklıkla tercih edilmektedir.

$$MAE(y,\hat{y}) = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i| \tag{3.9}$$

3.1.2.4. Çapraz Entropi (Cross Entropy - CE)

Çapraz Entropi fonksiyonu (Denklem 3.10), n tane istenilen değer (y) ile tahminlenen n tane sonucun ($\hat{y}$) logaritması çarpılarak elde edilir. Çapraz entropi fonksiyonu sınıflandırma gibi ayrık (discrete) değerlerin tahminlemesi problemlerinde tercih edilmektedir. Ayrıca çapraz entropi hata fonksiyonu kullanılan yapay sinir ağlarında çıkış katmanında Softmax veya Sigmoid aktivasyon fonksiyonları tercih edilmektedir.

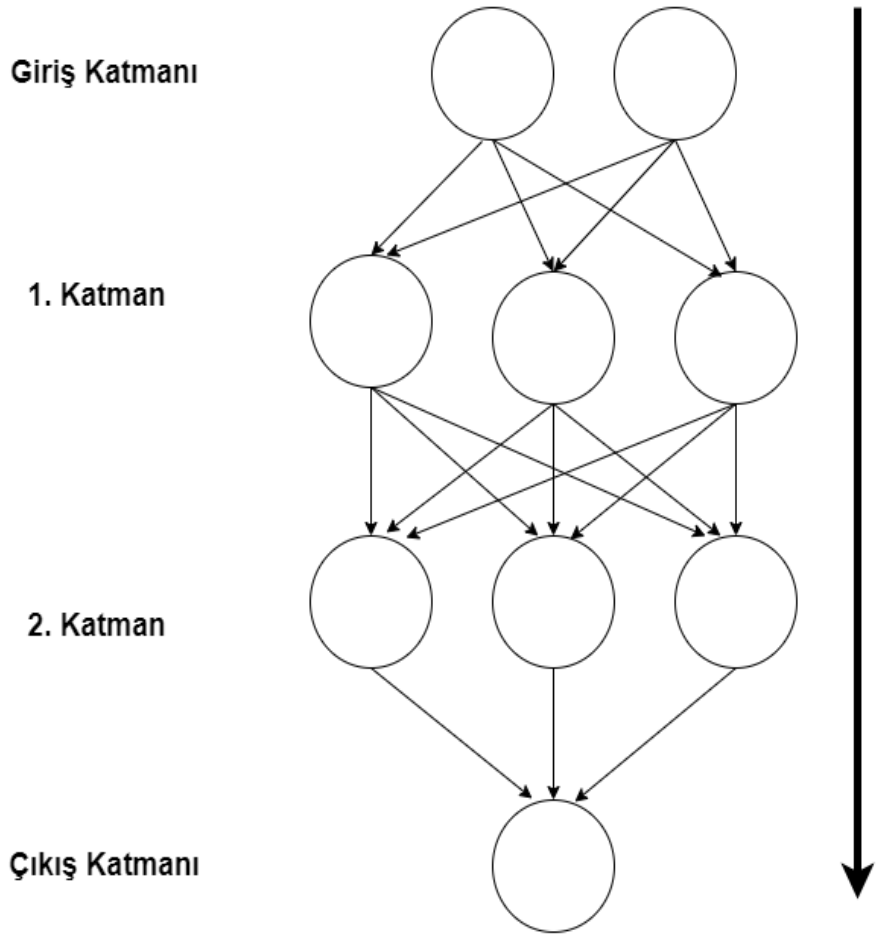
$$CE(y,\hat{y}) = -\sum_{i=0}^{n} y_i \log\hat{y}_i \; = \; -\sum_{i=0}^{n} -y_i \, log\hat{y}_i - (1-y_i)\log(1-\hat{y}_i) \tag{3.10}$$

3.2. İLERİ BESLEMELİ YAPAY SİNİR AĞLARI

İleri beslemeli yapay sinir ağları, girdi katmanından çıktı katmanına doğru bir yönde bilginin ilerlediği, döngüler olmayan bir tür yapay sinir ağıdır [29]. Bu ağlar, bilginin ileri doğru akışı nedeniyle ileri beslemeli yapay sinir ağları olarak adlandırılmaktadır. İleri beslemeli yapay sinir ağlarında, Şekil 3.6'da gösterildiği girdi katmanından çıktı katmanına doğru bilgi geri yüklenmeden aktarılır.

İleri beslemeli yapay sinir ağları, birbirlerine bağlı düğümlerden oluşan katmanlardan oluşmaktadır. İleri beslemeli yapay sinir ağlarında her katman, önceki katmandan gelen girdiyi işlemekte ve işlediği veriyi sonraki katmana göndermektedir. İleri beslemeli yapay sinir ağlarında girdi katmanı, ham girdi verisini almakta ve ilk gizli katmana (hidden layer) göndermektedir, bu katman veriyi işlemekte ve sonraki gizli katmana göndermektedir. Böylece son çıktı katmanı istenen çıktıyı üretmeye çalışmaktadır.

İleri beslemeli yapay sinir ağları, görüntü ve konuşma tanıma, doğal dil işleme ve zaman serisi tahmini gibi çeşitli problemlerde yaygın olarak kullanılmaktadır. Bu ağlar çeşitli optimizasyon algoritmaları kullanılarak eğitilebilmektedir. Ancak verideki karmaşık ilişkileri ve örüntüleri modelleme yetenekleri sınırlıdır ve öz yinelemeli bağımlılıkların kullanımını gerektiren zaman serisi veya bağlam bilgisinin kullanımını gerektiren alan transferi (domain transfer) gibi problemler için yetersiz kalmaktadır.

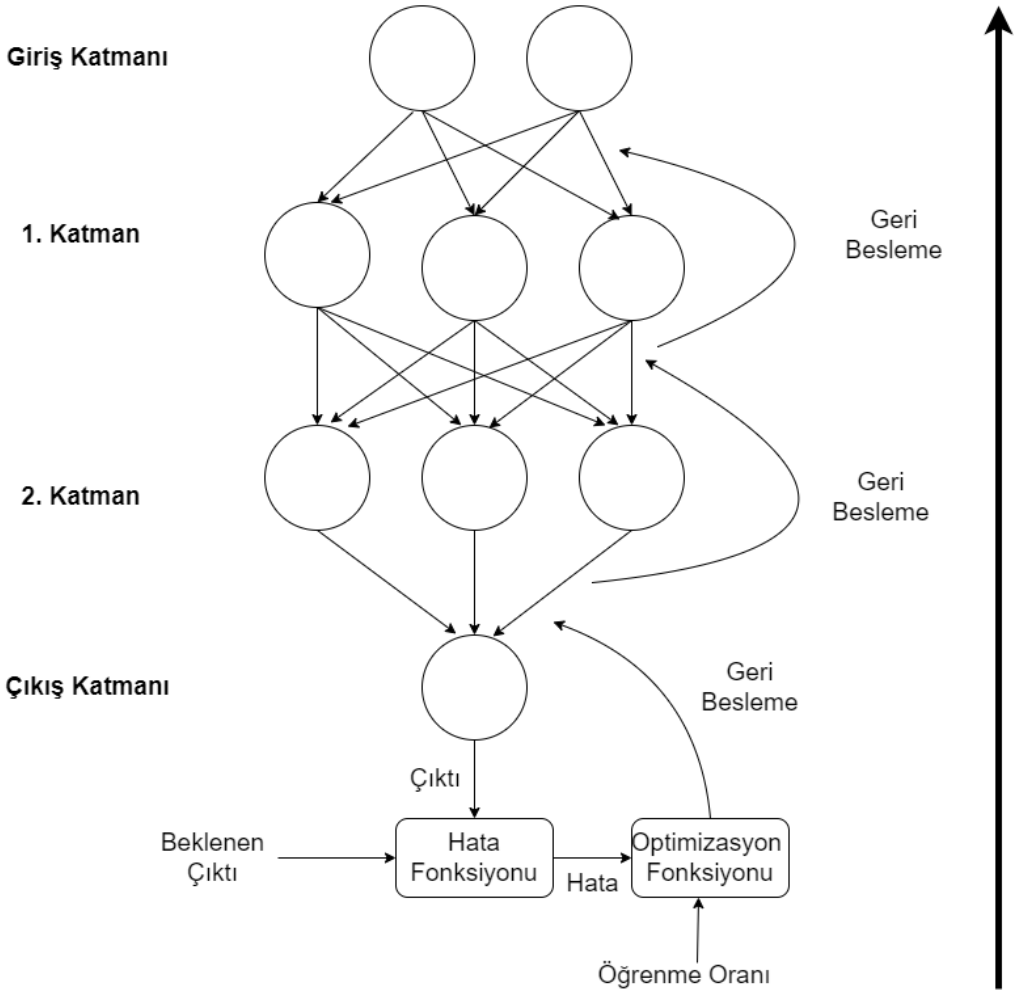


Şekil 3.6: İleri Beslemeli Ağ Yapay Sinir Ağı

3.3. GERİ YAYILIM ALGORİTMASI

Geri yayılım algoritması (backpropagation) yapay sinir ağlarının eğitiminde kullanılan bir yöntemdir. Geri yayılım algoritması, yapay sinir ağının giriş ve çıkış katmanları arasındaki ağırlık (weight) değerlerini güncellemek için kullanılmaktadır. Bu sayede yapay sinir ağının eğitimi yapılmaktadır.

Geri yayılım algoritması, Şekil 3.7'de gösterildiği gibi yapay sinir ağının eğitim verisiyle çalışmakta ve beklenen çıkış değerleriyle karşılaştırılmaktadır. Eğer yapay sinir ağının çıkış değerleri beklenen çıkış değerlerinden farklıysa, ağırlık değerleri güncellenmekte ve yapay sinir ağı tekrar eğitilmektedir. Bu süreç, beklenen çıkış değerlerine ulaşılana kadar devam ettirilmektedir. Bu süreç sonucunda yapay sinir ağının istenilen görev için eğitim işlemi tamamlanmaktadır.



Şekil 3.7: Geri Yayılım

Geri yayılım algoritması, öncelikle Denklem 3.11'de gösterildiği gibi y beklenen çıktısı ile $\hat{y}$ yapay sinir ağının çıktısı arasındaki E farkını/hatasını bölüm 3.1.1'de detaylıca anlatılan L hata fonksiyonları ile hesaplamaktadır.

$$E = L(y, \hat{y}) \tag{3.11}$$

Elde edilen E hatası daha sonra ağırlık değerlerinin güncellenmesi işlemi için kullanılmaktadır. W ağırlığını güncellemesi işlemi a öğrenme oranı (learning rate) ve E hatası kullanılarak Denklem 3.12'de gösterildiği şekilde yapılmaktadır.

$$W = W - a * \frac{\partial E}{\partial W} \tag{3.12}$$

Denklem 3.13'te gösterildiği gibi E hatası, L hata fonksiyonu ile y beklenen değeri ve $\hat{y}$ çıktısı kullanılarak hesaplanmaktadır. $\hat{y}$ çıktısı ise f aktivasyon fonksiyonu ile x girdi değeri ve b bias değeri için W ağırlığı kullanılarak hesaplanmaktadır. Denklem 3.13'te gösterildiği gibi E hatası direkt olarak W ağırlığı ile ilişkili değildir. Bu sebeple ağırlık güncellemesinde kullanılan Denklem 3.12'deki türev işlemi direkt olarak hesaplanamamaktadır. Bu problemi aşmak için Denklem 3.12'deki türev işlemi, zincir kuralı (Denklem 3.14) ile Denklem 3.15'te gösterildiği şekilde hesaplanmaktadır.

$$E = L(y, \hat{y}) \quad ve \quad \hat{y} = f(W * x + b) \tag{3.13}$$

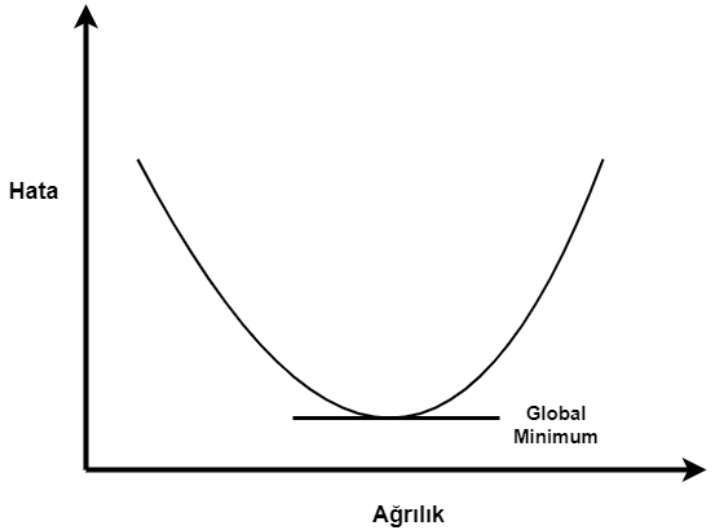
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} * \frac{\partial u}{\partial x} \tag{3.14}$$

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial W} \tag{3.15}$$

Bir derin öğrenme ağının en yüksek doğruluk ile çalışması için eğitim esnasındaki hatasının en az olması gereklidir. Bunun için de derin öğrenme ağının ağırlıkları ile elde ettiği çıktının hata değeri en az olması gerekmektedir. Hata fonksiyonun derin öğrenme modelini ağırlıkları ile elde edilen hata değerinin en az olduğu yer hata fonksiyonunun global minimum değeridir ve bu sebeple Şekil 3.8'de gösterildiği gibi geri yayılım algoritması ile hata fonksiyonun global minimumuna ulaşarak yapay sinir ağının en az hata ile çalışması hedeflenmektedir.

3.4. EVRİŞİMLİ SİNİR AĞLARI (CONVOLUTIONAL NEURAL NETWORKS - CNN)

Evrişimli sinir ağları genellikle görüntü işleme problemlerini çözmek için kullanılan bir derin öğrenme yaklaşımıdır. Evrişimli sinir ağları, görüntü işlemenin yanında ses tanıma gibi karmaşık örüntü tanıma problemlerinde de oldukça başarılı sonuçlar elde etmektedir. Evrişimli sinir ağlarındaki evrişim (convolution) katmanlarının yapay sinir ağlarına göre en büyük avantajı parametre sayısının azaltabilmesidir. Evrişimli sinir ağları getirdiği bu avantaj ile karmaşık problemleri çözmek için daha büyük ağların kurulmasına olanak sağlamaktadır [29]. Evrişimli sinir ağları farklı filtreler yardımıyla girdi (input) verisi üzerinden, girdi verisinin probleme yönelik ve/veya genel özniteliklerini (feature) çıkartır.



Şekil 3.8: Hata - Ağırlık Eğrisi

Öznitelik çıkartma işlemi sonucunda çıkartılan öznitelikler, derin öğrenme ağı derinleştikçe daha karmaşık ve daha ayırt edici öznitelikler bulabilmektedir. Örneğin bir resim girdisi için ilk evrişim katmanlarında resimdeki yatay ve dikey çizgileri bulmayı öğrenirken daha sonraki katmanlardaki evrişim katmanları, resimdeki kenar çizgilerini bulmayı öğrenebilmektedir. Bununla birlikte derin öğrenme ağının derinleştirilmesi, geliştirilen derin öğrenme ağındaki parametre sayısını arttırmaktadır. Bu da derin öğrenme ağındaki parametrelerin bulunması (tuning) için daha fazla veri ihtiyacı doğurmaktadır. Eğer geliştirilen derin öğrenme ağı, yeterli miktarda veri ile eğitilmez ise yapılacak eğitiminin sonucunda öğrenme yerine sadece kullanılan verilere iyi tepki verme problemi olan aşırı öğrenme (overfitting) veya yetersiz öğrenme (underfitting) problemlerinin yaşanması muhtemeldir. Bu nedenle derin öğrenme

algoritmalarında olduğu gibi evrişimli sinir ağlarında da verinin miktarı ve çeşitliliği başarılı bir eğitim (train) işlemi için oldukça önemlidir.

Evrişimli sinir ağları veriden öznitelik çıkarmak için birçok farklı katman içermektedir. Bu katmanlardan en popülerleri devam eden bölümlerde detaylıca anlatılmıştır.

3.4.1. Evrişim Katmanı (Convolution)

Evrişim katmanı, evrişimli sinir ağlarının en temel katmanıdır. Resim verisi, $görüntünün\ yüksekliği\ \times\ görüntünün\ genişliği\ \times\ görüntünün\ kanalı\ (channel)$ boyutlu bir tensördür. Elde edilen tensörün kanal değerleri girdinin RGB, CMYK, HSV gibi renk uzayları ile ilişkilidir. Ancak bir resim verisi genelde RGB adı verilen 0-255 değer aralığında değer alan resmin kırmızı, yeşil ve mavi değerleri ile ifade edilmektedir. Evrişim katmanı resim girdisi üzerinde her kanal için verinin boyutuna ($yükseklik\ \times\ genişlik$) eşit veya verinin boyutundan daha küçük boyutta filtreler oluşturup veriden veriye özgü öznitelik çıkartmaya çalışır. Çıkartılan öznitelikler evrişimli sinir ağı derinleştikçe karmaşıklaşmaktadır. Evrişim katmanı bu işlemi evrişim işlemi ile yapmaktadır. $m\ \times\ n$ boyutlu bir f resim matrisi ve $j\ \times\ k$ boyutlu bir h fitresi için evrişim işlemi Denklem 3.16'da ve Şekil 3.11'de gösterilmiştir.

$$G[m,n] = (f * h)[m,n] = \sum_{j} \sum_{k} h[j,k] f[m-j, n-k] \tag{3.16}$$

3.4.2. Yığın Normalizasyonu (Batch Normalization - BN)

Eğittim esnasında yapay sinir ağları hata fonksiyonları ile hesaplanan hata değeri kullanılarak geri yayılım (BackPropagation) algoritması ile ağırlıklarını günceller ve öğrenmeye başlar. Bu süreçte her katman kendi hatasını sırayla ve kendisinden bir önceki katmandan geri beslenen değerleri de kullanarak minimize etmeye çalışır. Bu da bir katmandaki parametrelerin değişikliği sonraki katmanların parametrelerinde de değişimine sebep olmaktadır. Bu sıralı öğrenim (örneğin 4. katmandaki öğrenme süreci başlaması için 3. katmandaki öğrenme süreci bitmesi gerekmektedir) iç ortak değişken kayması (internal covariate shift) adı verilen probleme sebep olmaktadır. İç ortak değişken kayması problemini azaltmak için Ioffe ve Szegedy [30] yığın normalizasyon tekniğini önerdiler. Yığın normalizasyonu ayrıca katmanlar arasındaki girdi-çıktı dağılımlarının ortalamasını 0, standart sapmasını 1 olacak hale getirmektedir. Bu işlemi n boyutlu x girdisi için yapmak amacıyla öncelikle Denklem 3.17 ile μ ortalaması ve Denklem 3.18 ile σ standart sapması hesaplanmakta daha sonra elde edilen değeler ile Denklem

3.19 ile normalize edilmiş $\hat{x}$ hesaplanmaktadır. Son olarak Denklem 3.20'de gösterildiği gibi elde edilen $\hat{x}$ değerini ölçeklemek (scale) için γ katsayısı ile çarpılmakta ve kaydırmak (shift) için β katsayısı ile toplanmaktadır.

$$\mu_x = \frac{1}{n}\sum_{i=0}^{n} x_i \tag{3.17}$$

$$\sigma_x = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - \mu_x)^2} \tag{3.18}$$

$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \tag{3.19}$$

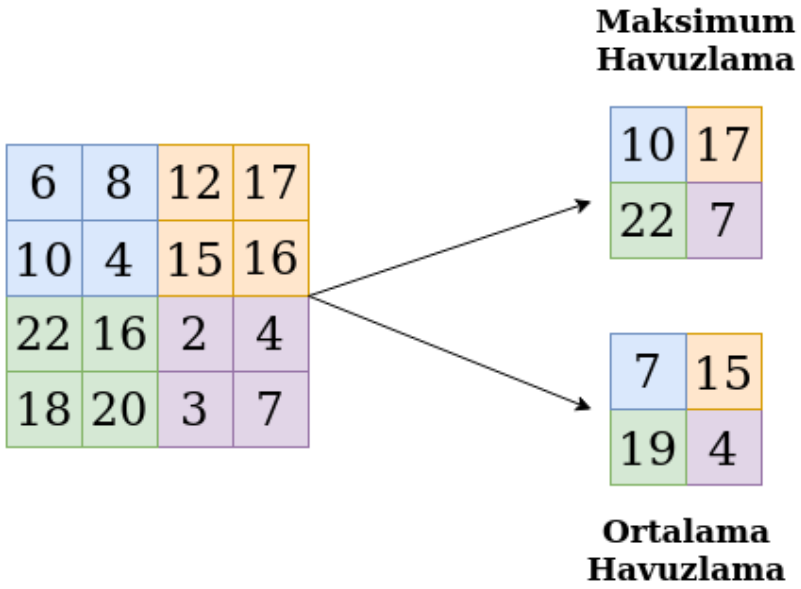
$$y = (\gamma * \hat{x}) + \beta \tag{3.20}$$

3.4.3. Havuzlama Katmanı (Pooling)

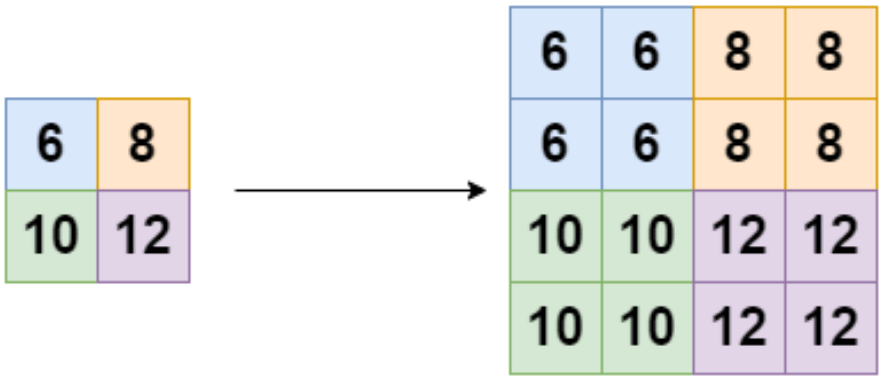
Havuzlama katmanı kendinden önceki katmanın çıktısının boyutunu, havuzlama tipine göre azaltmak için kullanılmaktadır. Havuzlama katmanı (Şekil 3.9) maksimum ve ortalama olmak üzere 2 tip olarak kullanılmaktadır. Maksimum havuzlama katmanı, girdi üzerinde gezdirilen çekirdeğin (kernel) havuzlayacağı örneklem değerlerinin sadece en büyük değerini, ortalama havuzlama katmanı ise bütün değerlerin ortalamasını çıktı olarak vermektedir. Böylece bir önceki katmanın boyutu küçültülerek sadece baskın değerlerin bir sonraki katmana geçirilmesini sağlar ayrıca maksimum ve ortalama havuzlama katmanları girdi verisindeki gürültüleri de engelleyebilmektedir. Havuzlama katmanı bir önceki katmanın çıktı verisinin boyutunu küçülterek bir sonraki katmana verdiği için ağın hesaplama gücü (compute power) miktarını ve süresini azaltmaktadır.

3.4.4. Üst Örnekleme Katmanı (UpSampling)

Üst örnekleme katmanı, havuzlama katmanın tersi şeklinde görev yapar. Üst örnekleme katmanı, Şekil 3.10'da gösterildiği gibi bir önceki katmanın çıktı verisinin boyutunu büyütmekte kullanılır. Üst örnekleme katmanında, görüntü işleme algoritmalarında üst örnekleme işlemi için kullanılan interpolasyon metotları kullanılmaktadır. Üst örnekleme katmanı Kodlayıcı - Kod Çözücü mimarilerinde sıklıkla tercih edilmektedir.



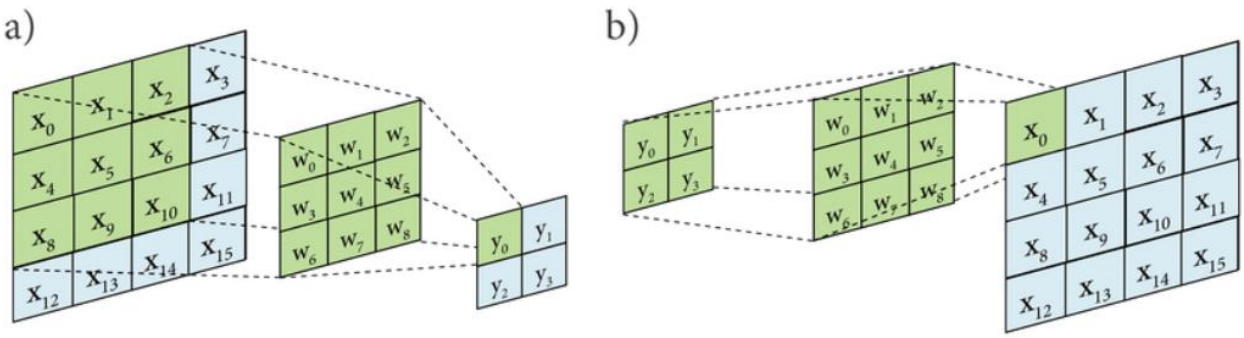
Şekil 3.9: Havuzlama Katmanı



Şekil 3.10: Üst Örnekleme

3.4.5. Devrik Evrişim Katmanı (Transposed Convolution)

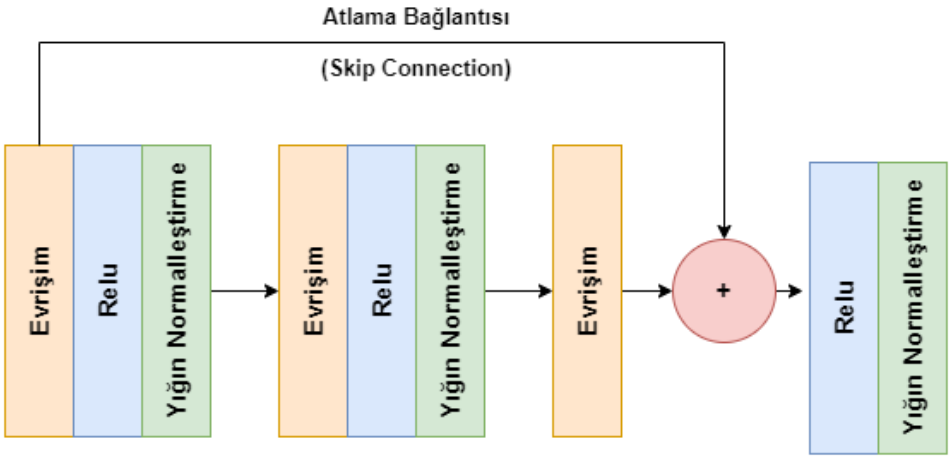
Devrik evrişim katmanı (Şekil 3.11.B), üst örnekleme katmanı ile aynı işlevi yapmak için kullanılmaktadır. Ancak boyut büyütme işlemi yaparken evrişim metodu ile yapmaktadır. Devrik evrişim katmanı, üst örnekleme katmanında olduğu gibi Kodlayıcı - Kod Çözücü mimarilerinde sıklıkla tercih edilmektedir.



Şekil 3.11: (A) Evrişim işlemi (B) Devrik evrişim işlemi [19]

3.5. ARTIK AĞ (RESIDUAL NEURAL NETWORK - RESNET) YAKLAŞIMI

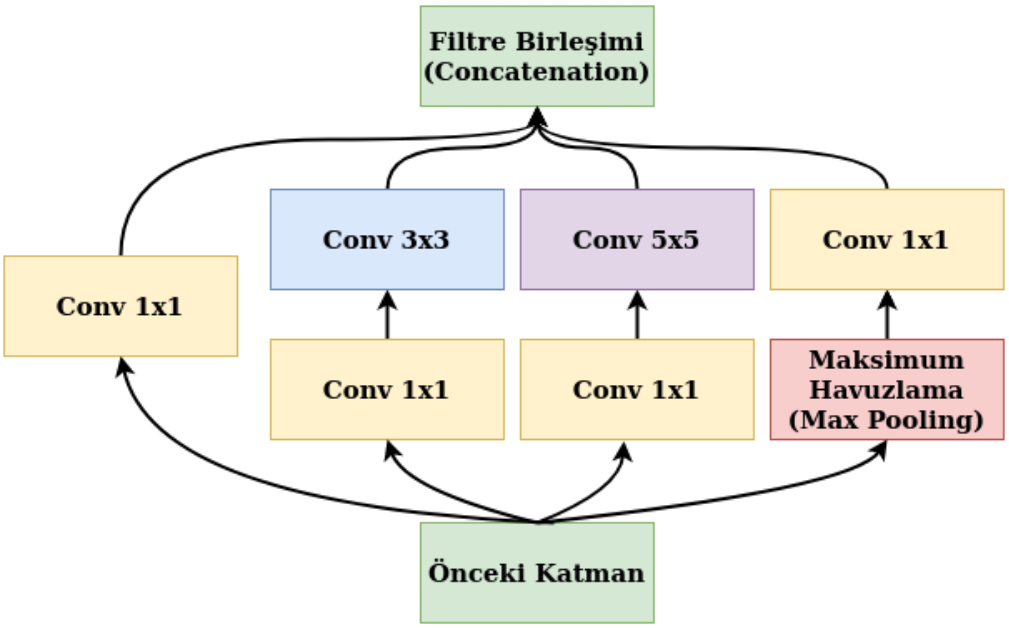
Derin öğrenme algoritmalarında kurulan çok derin katmanlı mimarilerde eğitim işlemi esnasında, Denklem 3.12'de gösterilen gradyanların kaybolması (degradation) problemi yaşanmaktadır. Ağ derinleştikçe, yeterli sayıda veri ile de eğitim yapılıyorsa daha az derin ağın performansının daha derin ağın performansından düşük olması beklenir. Ancak gerçekte ağ derinleştikçe degradiation problemi sebebiyle daha az derin olan ağ, daha derin olan ağa göre daha yüksek performans gösterebilmektedir. Bu problemin giderilmesi için He ve diğ. [21] artık ağ yaklaşımını önerdiler. Artık ağ yaklaşımında bir katman, kendisinden daha önceki katman veya katmanlar ile aradaki katmanlar es geçilerek (örneğin 5. katman ile 3. katman) direkt olarak bağ kurması (Şekil 3.12) ile sağlanır. Bu işlem degradation probleminin önüne geçilmesini ve derin ağların daha performanslı çalışmasını sağlamaktadır.



Şekil 3.12: Atlama Bağlantısı

3.6. AĞ İÇİNDE AĞ YAKLAŞIMI VE INCEPTION MODÜLÜ

Görüntü sınıflandırma problemlerinde, girdi verilerindeki sınıfa ait piksellerin görüntü üzerinde kapladığı alan bazı girdilerde fazla olurken bazı girdilerde daha az olabilmektedir. Sınıfına ait bilgilerin girdi verisinde kapladığı alanın farklılığı nedeniyle evrişim işlemi yapılırken filtrelerin çekirdek boyutunu en optimum şekilde ayarlamak, girdi verisinden öznitelik çıkarımı için oldukça önemlidir. Büyük öznitelikler için büyük çekirdek boyutu, daha küçük öznitelikler için daha küçük çekirdek boyutlu filtreler ile evrişim işlemi yapılmalıdır. Ancak büyük boyutlu çekirdekli filtreler hesaplama açısından maliyetli olmaktadır. Büyük boyutlu çekirdek filtrelerin hesaplama maaliyetinin düşürülmesi için Lin ve diğ. [32] girdi verisinin önce n filtre sayısı olmak üzere $1 \times 1 \times n$ çekirdek boyutlu filtre ile evrişim yapılmasını ve daha sonra $k \times l \times m$ çekirdek boyutlu filtreler kullanılmasını önerdiler. Bu sayede girdi kanalları 1×1 kernel boyutlu evrişim işlemi ile önce girdinin boyutu istenilen boyutlara daha düşük hesaplama maaliyetleri ile ulaşılmaktadır. Szegedy ve diğ. [33] ise *inception* adını verdikleri, 1×1 çekirdek boyutlu evrişim işlemi yapıldıktan sonra farklı boyuttaki özniteliklerin çıkarımı işlemi için aynı seviyede farklı çekirdek boyutlu filtreler kullanarak öznitelik çıkartma metodunu önerdi. Bu sayede hem aynı katman seviyesinde birden fazla boyutta çekirdek kullanılarak öznitelik çıkarımı işlemi yapılmakta hem de parametre sayısında ciddi bir azalma meydana gelmektedir. Inception modülünün 1. versiyonu Şekil 3.13'deki gibidir.

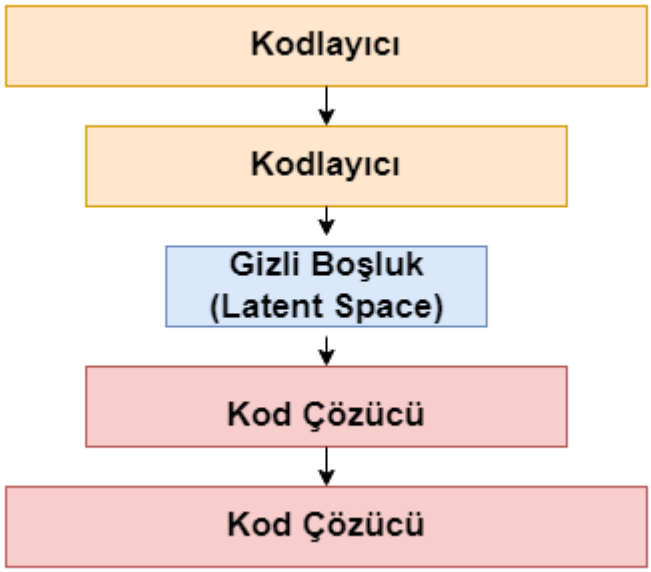


Şekil 3.13: Inception V1 Modülü

3.7. KODLAYICI – KOD ÇÖZÜCÜ (ENCODER - DECODER)

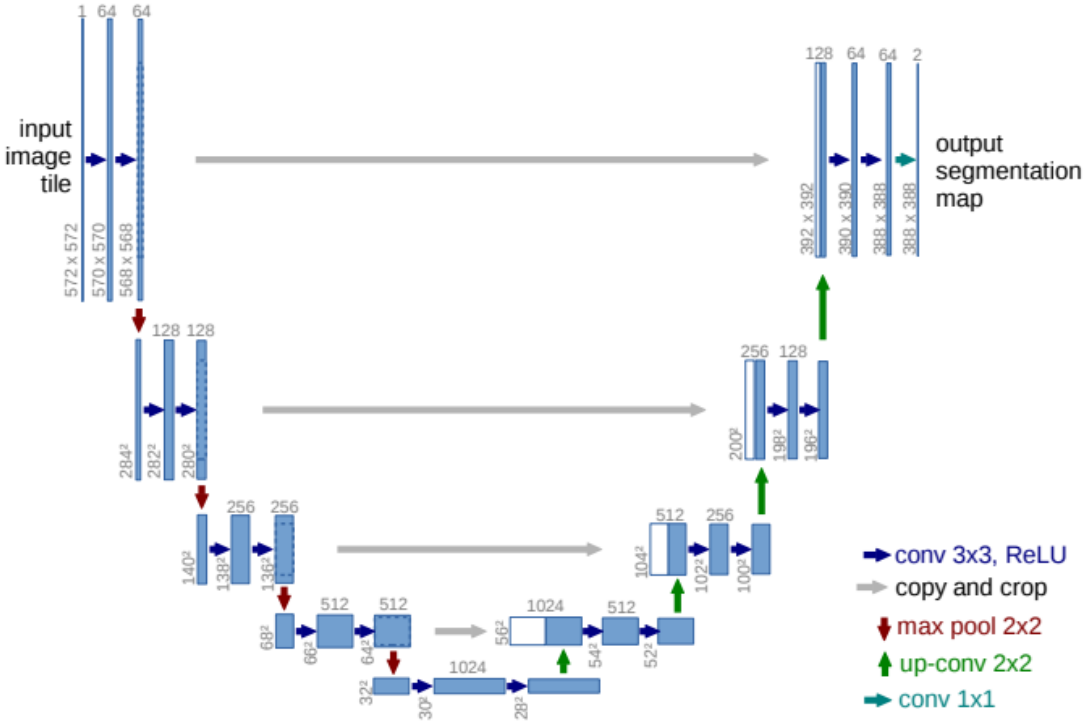
Kodlayıcı - kod çözücü yaklaşımı, girdileri istenilen amaca uygun manipüle edilip tekrar oluşturan derin öğrenme yaklaşımıdır. Kodlayıcı - kod çözücü medikal görüntülerde anlamsal bölütleme, gürültü filtreleme gibi uygulamalarda oldukça iyi sonuç vermektedir.

Kodlayıcı - kod çözücü yaklaşımı Şekil 3.14'de gösterildiği gibi öncelikle girdi verilerinin boyutları havuzlama veya evrişim işleminde adımlama (stride) gibi yöntemler ile azaltılmakta ve öznitelikler sıkıştırılmaktadır (Kodlama - Encode). Bu yapılan işlem ile güçlü (robust) ve baskın öznitelikleri ön plana çıkarılmaktadır. Ön plana çıkarılan öznitelikler daha sonra üst örnekleme veya devrik evrişim katmanları ile kod çözme işlemi yapılan katmanların çıktısının boyutu artırılmaktadır (Kod çözme - Decode).



Şekil 3.14: Kodlayıcı - Kod Çözücü Mimarisi

Bu sayede girdi verisinin baskın öznitelikleri ile çözülmek istenen probleme göre girdi verisi manipüle edilebilmektedir. Kodlayıcı – kod çözücü yaklaşımının en yaygın örneği U-Net [34] mimarisidir (Şekil 3.15). U-Net mimarisinde kodlayıcı katmanlar aynı boyuttaki kod çözücü katmanlarına atlamalı bağlantılar yapılmaktadır. Bu işlem sayesinde girdi verisinin öznitelikleri ve anlamsal bütünlükleri kod çözücü katmanlara da iletilmektedir. Kodlayıcı - kod çözücü yaklaşımı anlamsal bölütleme (semantic segmentation), alan transferi (domain transfer), çözünürlük artırımı, gürültü filtreleme gibi görüntü manipülasyonlarının yapıldığı uygulamalarda sıklıkla kullanılmaktadır.

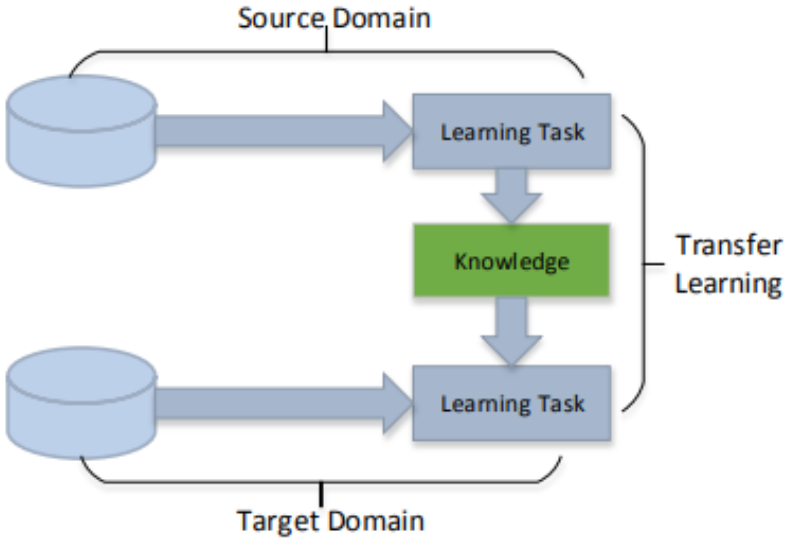


Şekil 3.15: U-Net [34]

3.8. ÖĞRENME AKTARIMI (TRANSFER LEARNING)

Öğrenme aktarımı yöntemi, daha önce başkaları tarafından belli bir amaca yönelik geliştirilen ve eğitilen (pretrained) bir ağın daha sonra aynı veya farklı bir problemde öznitelik çıkarımı işlemi yapılması için kullanılmaktadır. Öğrenme aktarımı yönteminde kullanılacak daha önceden eğitilen derin öğrenme ağı tekrar eğitim yapılmasına ihtiyaç duymamaktadır. Öğrenme aktarımı yönteminde (Şekil 3.16) daha önceden öğrenilmiş bilgi ve öznitelik çıkartma aşamaları kullanılarak yeni derin öğrenme ağının farklı veri kümesi üzerinde daha iyi performans göstermesi sağlanmaktadır [35].

Elimizde N_1 ve N_2 olmak üzere 2 derin öğrenme ağı ve D_1 ve D_2 olmak üzere 2 veri kümesi olsun. Öğrenme aktarımındaki ana hedef N_1 derin öğrenme ağının D_1 veri kümesinden elde ettiği bilgiyi ve deneyimi kullanarak N_2 derin öğrenme ağının D_2 veri kümesi üzerindeki başarısını arttırmaktır.



Şekil 3.16: Öğrenme Aktarımı [35]

Öğrenme aktarımı yöntemi ile bütün derin ağı eğitmek yerine sadece problemi çözmesi beklenen katmanların eğitimi yapılması sağlanır ve optimize edilecek parametre sayısı çok aza indirilir. Bu sayede hem ağı eğitmek için gerekli olan veri miktarında hem de hesaplama gücünde büyük ölçüde azalma meydana gelmektedir.

3.9. 3D YENİDEN OLUŞTURMA (3D RECONSTRUCTION)

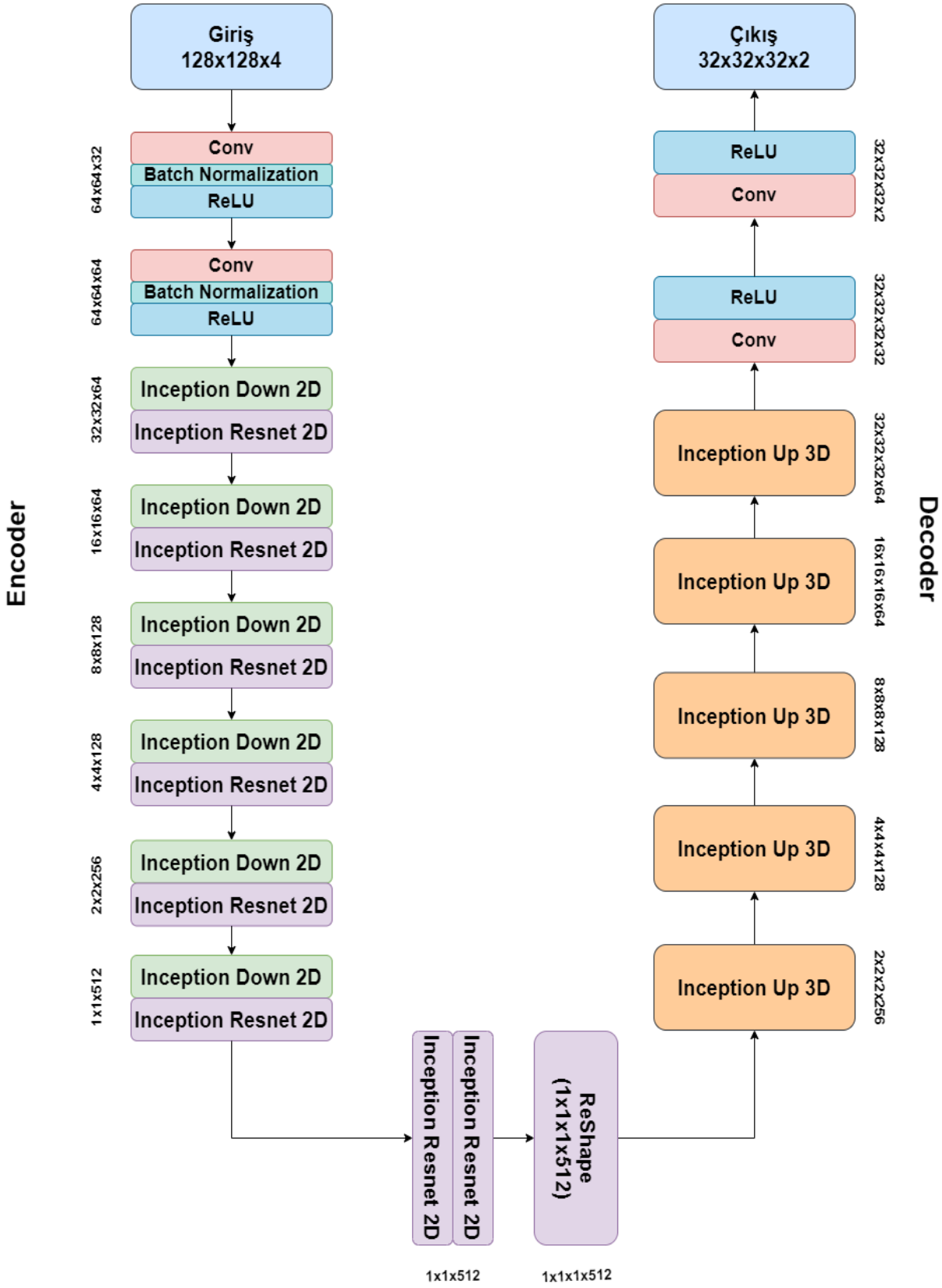
Tez konusu olan 3D yeniden oluşturma için geliştirilen derin öğrenme modelinde kullanılmak üzere tez çalışması kapsamında, Inception Alt Örnekleme 2D (Şekil 3.21), Inception ResNet 2D (Şekil 3.22) ve Inception Üst Örnekleme 3D (Şekil 3.23) olmak üzere 3 alt ağ modülü geliştirilmiştir ve bu 3 alt ağ modülü kullanılarak derin öğrenme ağı tasarlanmıştır. Önerilen derin ağ modelinde (Şekil 3.17), 2 boyutlu resimlerdeki güçlü ve dominant öznitelikleri daha belirgin hale getirmek ve belirginleşen öznitelikleri kullanarak 3. boyuttaki koordinat verilerini oluşturmak (3D reconstruction) için Encoder - Decoder yaklaşımı tercih edilmiştir. Bununla beraber önerilen derin öğrenme ağındaki alt modüllerde parametre sayısını azaltmak ve farklı boyutlardaki öznitelikleri yakalamak için inception yaklaşımı tercih edilmiştir. Ayrıca derin ağlarda oluşan gradyan kaybolması probleminin önüne geçmek için ağın "Encoder" kısmında ResNet yaklaşımı kullanılmıştır. Ağın "Decoder" kısmında UpSampling işlemi için devrik evrişim işlemi tercih edilmiştir. Tez konusu olan 3D yeniden oluşturma için geliştirilen bütün alt ağ modülleri ve yaklaşımların sonunda elde edilen ve önerilen derin ağ modeli toplamda

2.529.282 adet parametre içermektedir. Geliştirilen derin ağ modelinin eğimi için ShapeNet [26] veri kümesindeki 2000 ve üzeri sayıda veri içeren kategoriler kullanılmıştır. Bu eleme işlemi yapıldığında 7 kategoride toplam 34691 farklı veri elde edilmiştir. Toplanan verilerin %60'ı (20.814) eğitim, %20'si (6.938) validasyon, %20'si (6.939) ise test verisi olarak kullanılmıştır. Oluşturulan veri kümesi kategorik olarak denetimsiz (unsupervised) şekilde kullanılmıştır.

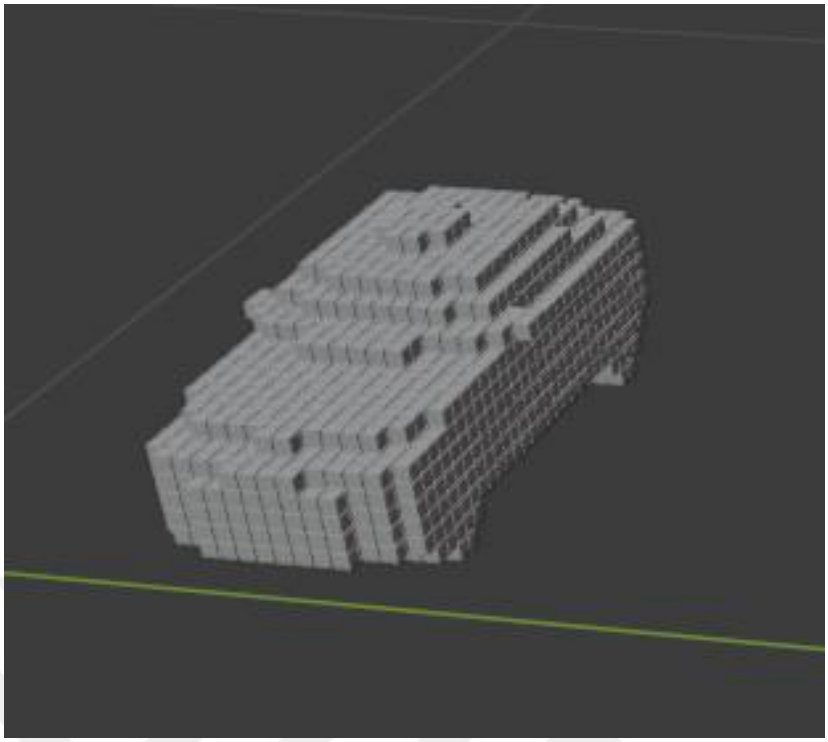
Derin öğrenme ile 3 boyutlu yeniden oluşturma problemi için hem tek açıdan çekilen görüntüler kullanarak [10, 15-17] hem de birden çok açıdan çekilen görüntüler kullanarak [16, 17] 3D modeller üretebilen çalışmalar mevcuttur. Ancak bu çalışmalar kategorik olarak denetimli olacak şekilde geliştirilmiştir. Bunun yanında Gwak ve diğ. önerdiği McRecon [16] ve Yan ve diğ. önerdiği PTN [17] yöntemleri ShapeNet veri kümesini kullanarak Voxel tabanlı 3 boyutlu model çıktısı üretebilmektedir. PTN ve McRecon yöntemlerinin hem girdi sayısı olarak 5 görüntü kullandığı hem de girdi sayısı olarak 1 görüntü kullandığı versiyonları mevcuttur. Hem PTN hem de McRecon yönteminde de girdi sayısı olarak 5 görüntü kullandığı versiyonlar, girdi sayısı olarak 1 görüntü kullandığı versiyonlara göre daha iyi çıktı üretmektedir. Üretilen çıktıların başarısı bölüm 5'de detaylıca anlatılmış ve tartışılmıştır.

3.9.1. Veri Ön İşleme (Preprocessing) ve Veri Kümesi

Tez konusu olan 3D yeniden oluşturma için oluşturulan derin öğrenme modelinde kullanılmak üzere tez çalışması kapsamında ShapeNet veri kümesi kullanılmıştır. ShapeNet veri kümesinde her veri için 24 adet farklı açıdan çekilmiş 137×137 boyutlu RGB renk uzayında resimler ve bu resimlere karşılık olarak Voxel gösterimli 3D modeller mevcuttur. ShapeNet veri kümesindeki 1 Voxel gösterimli 3D model Şekil 3.18'de ve Voxel gösterimli 3D modelin karşılığı olan 24 adet 2D RGB görüntüler Şekil 3.19'da gösterildiği gibidir. Eğitim verisi hazırlanırken veri kümesindeki 24 açıdan sadece 4 açıdan çekilmiş görseller ve bu görsellere karşılık voxel gösterimli 3D model kullanılmıştır. Eğitim verisi için kullanılan 4 açıdan görsel, veri kümesinde bulunan Voxel gösterimli 3D modellerin karşılığı olan 24 farklı açıdan çekilmiş görseller içerisinden başlangıcı rastgele olmak üzere ardışık olarak seçilmiştir.



Şekil 3.17: Önerilen Model



Şekil 3.18: ShapeNet Veri Kümesindeki Örnek 3D Model

Eğitim verisinde kullanılan 4 farklı açıdan alınan her bir görüntü önce R kırmızı pixel değeri, G yeşil pixel değeri ve B mavi pixel değeri için International ITU-R BT.601-7 (03/211) standardında[1] belirtilen Denklem 3.21 ile Gri renk değeri üretilip, RGB renk paletinden gri renk paletine çevrilmiştir.

$$Gri(R,G,B) = \big((0.299) * R + (0.587) * G + (0.114) * B\big) \tag{3.21}$$

Gri renk paletine çevrilen görüntüler, M görüntü matrisinin i. satır ve j. sütunundaki değeri için t eşik değeri kullanılarak eşikleme yöntemi (Denklem 3.22) kullanılarak resimlerin siluet görüntüleri elde edilmiştir.

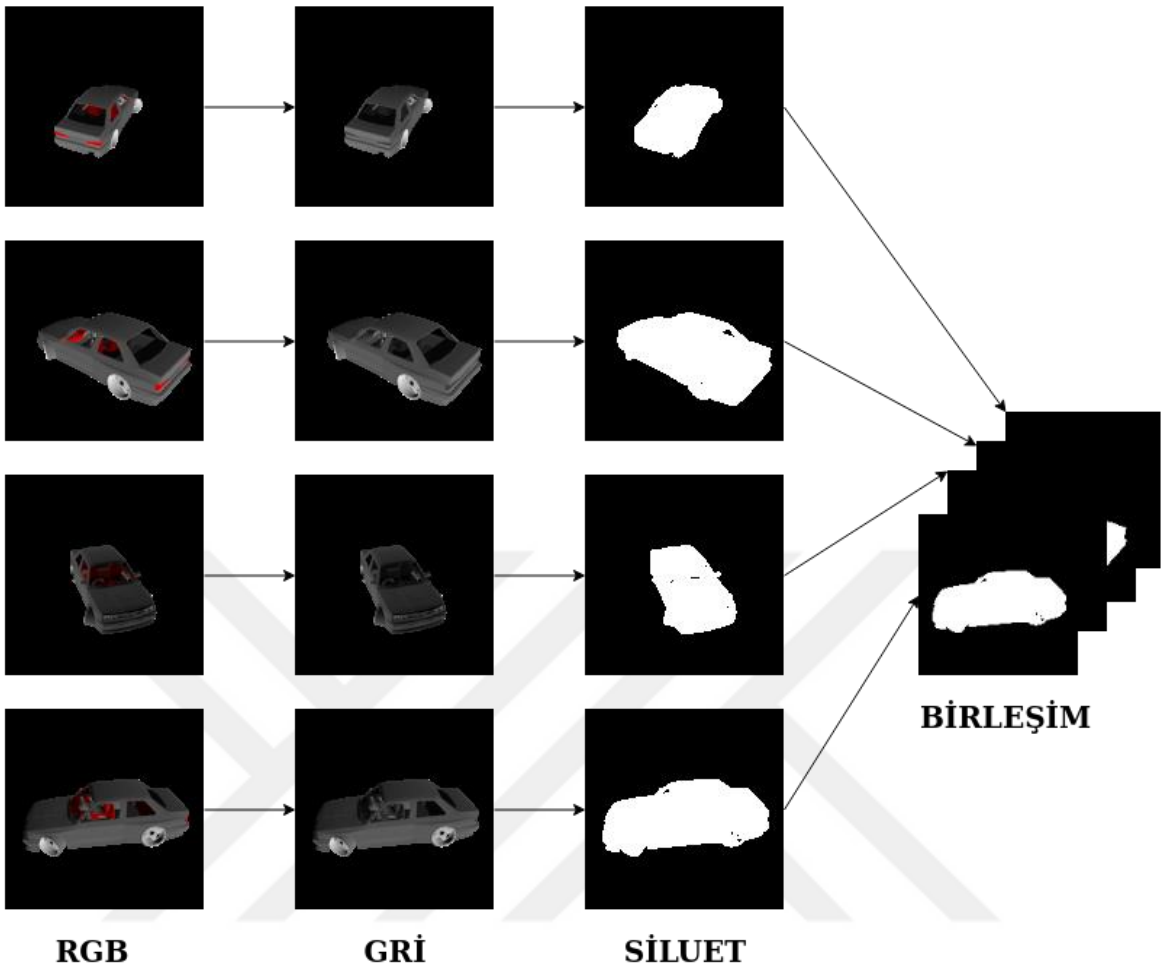
$$T(t,i,j) = \begin{cases} 0, & M[i,j] < t \\ 255, & M[i,j] \geq t \end{cases} \tag{3.22}$$

Daha sonra 137×137 boyutundaki görüntüler 128×128 boyutuna indirgenmiştir. En son 4 farklı açıdan çekilmiş görüntülerin siluet görüntüleri kanal olarak birbirleri üzerine

[1] https://www.itu.int/rec/R-REC-BT.601

eklenmiştir. Elde edilen son görüntünün boyutu $128 \times 128 \times 4$ şeklindedir. Veri önişleme adımları Şekil 3.20'de gösterilmiştir. Elde edilen son görüntünün 255'e bölünerek normalize edilmiş ve görüntülerin pixel değer aralığı 0-255'ten, 0-1 değer aralığına çekilmiştir. Bu işlemden sonra veri ön işleme adımları sonlanmaktadır.

Şekil 3.19: ShapeNet Veri Kümesindeki Örnek 2D Görüntüler



Şekil 3.20: Veri Ön İşleme

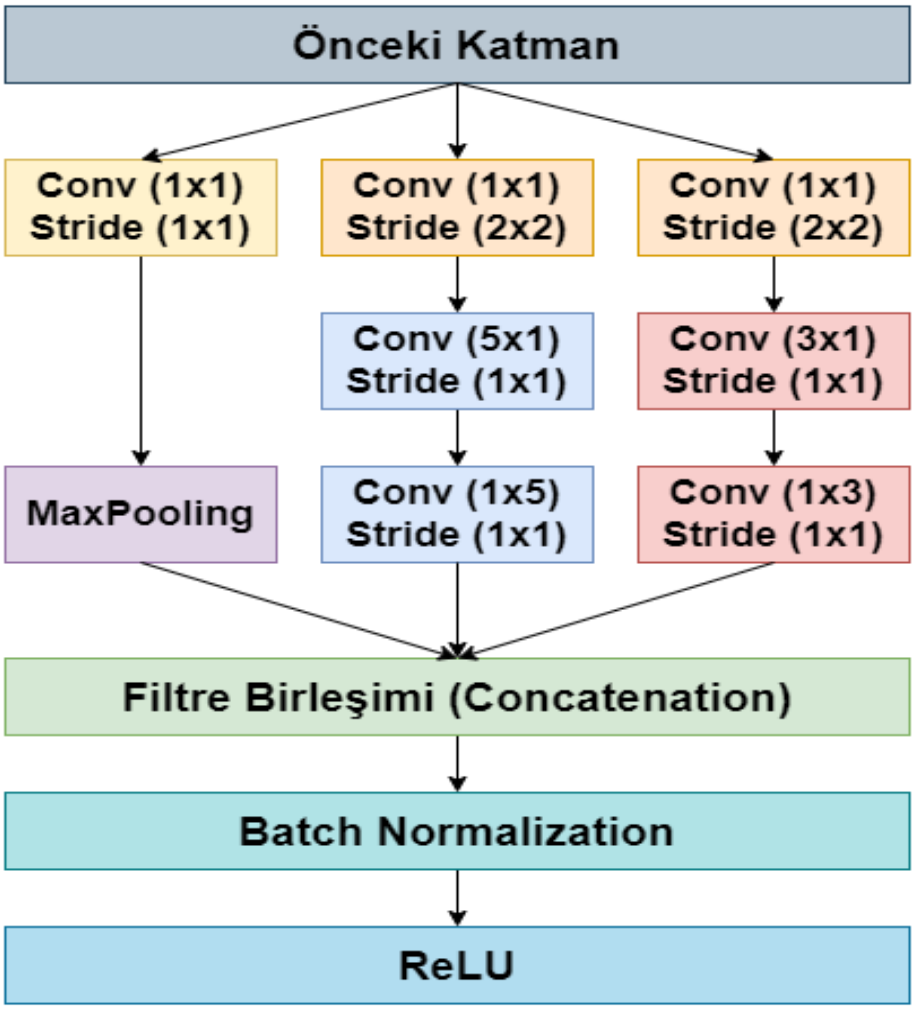
3.9.2. Inception Alt Örnekleme (DownSampling) 2D Ağ Modülü

Inception Alt Örnekleme 2D ağ modülü (Şekil 3.21), 2 boyutlu girdilerin yükseklik ve genişlik eksenlerinin boyutlarının azaltılması ve kanal sayısının artırılması ile öznitelik çıkarmak için geliştirilmiştir. Inception Alt Örnekleme 2D ağ modülünde öncelikle orijinal inception yönteminin önerildiği çalışmada [33] gösterildiği gibi 1×1 kernel boyutlu evrişim işlemi ve ardından aynı katman seviyelerinde 3×1 ve 1×3 boyutlu, 5×1 ve 1×5 boyutlu çekirdeklerin kullanıldığı evrişim ve Max Pooling katmanları kullanılmıştır.

Inception Alt Örnekleme 2D ağ modülünde alt örnekleme (down sampling) yapmak amacıyla hem Max Pooling katmanı hem de 1×1 kernel boyutlu evrişim işlemlerinde 2×2 stride (adım) kullanılmıştır. Daha sonra elde edilen filtreler birleştirilerek (concatenation) sırası ile yığın normalleştirme (Batch Normalization) ve Relu aktivasyon fonksiyonu katmanlarında geçmektedir. Her Inception Alt Örnekleme 2D ağ modülünde en az parametre ve farklı kernel

boyutlarında en fazla filtre ile bir önceki katmandan gelen girdinin boyutu 2×2 küçülmesi amaçlanmıştır. Örneğin bir önceki katmandan gelen girdinin boyutu F kanal sayısı olmak üzere $32 \times 32 \times F_1$ ise, Inception Alt Örnekleme 2D ağ modülünden sonra bu boyut $16 \times 16 \times F_2$ olacaktır.

Inception Alt Örnekleme 2D ağ modülün çıktısında, hem bir önceki katmandan gelen girdinin boyutu yarıya indirilmekte hem de yarıya indirilen girdilerin, farklı boyutta çekirdekler (kernel) kullanılarak yapılan evrişim işlemleri sayesinde farklı boyutlardaki özniteliklerin çıkarılması hedeflenmektedir.



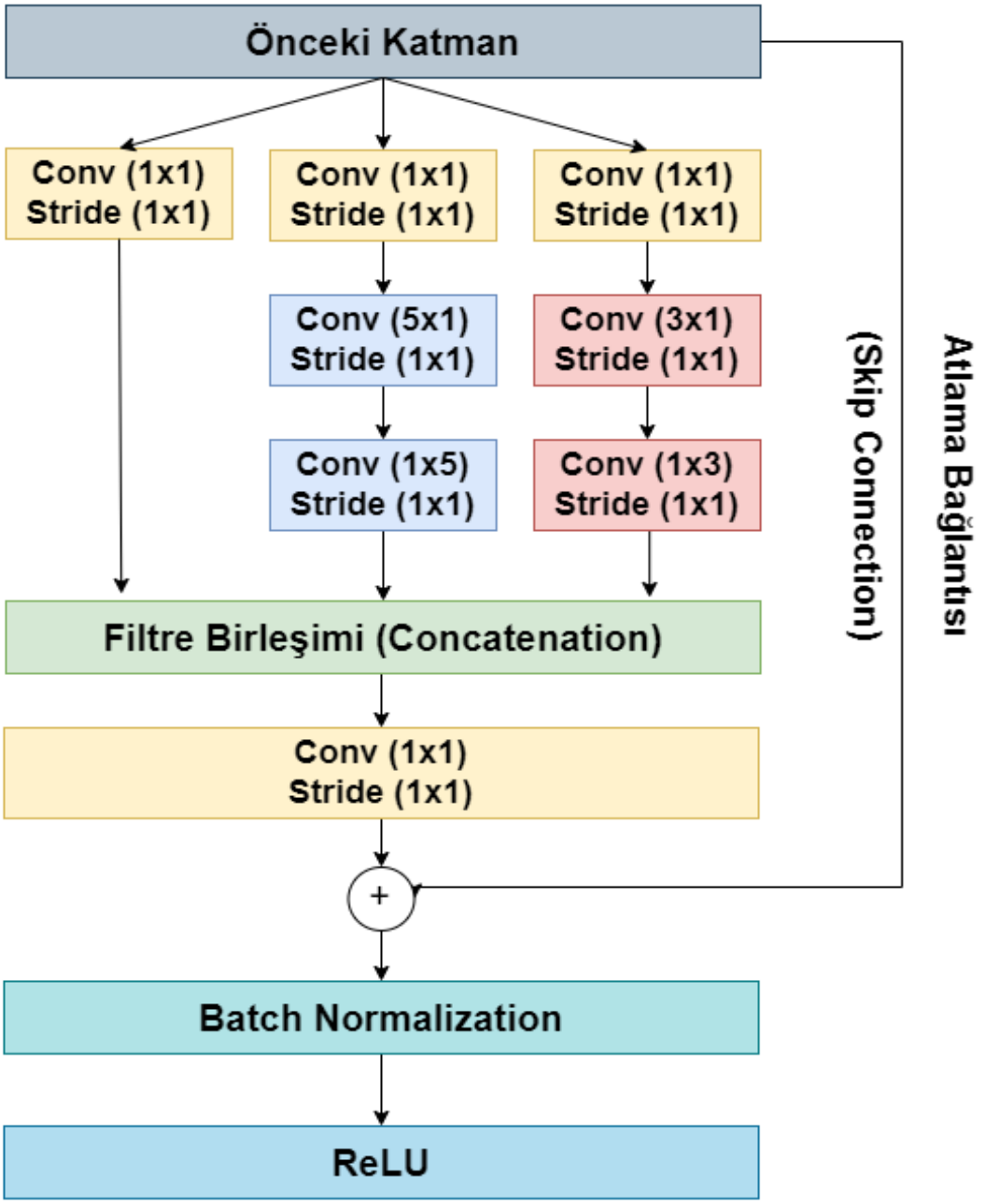
Şekil 3.21: Inception Alt Örnekleme 2D Ağ Modülü

3.9.3. Inception ResNet 2D Ağ Modülü

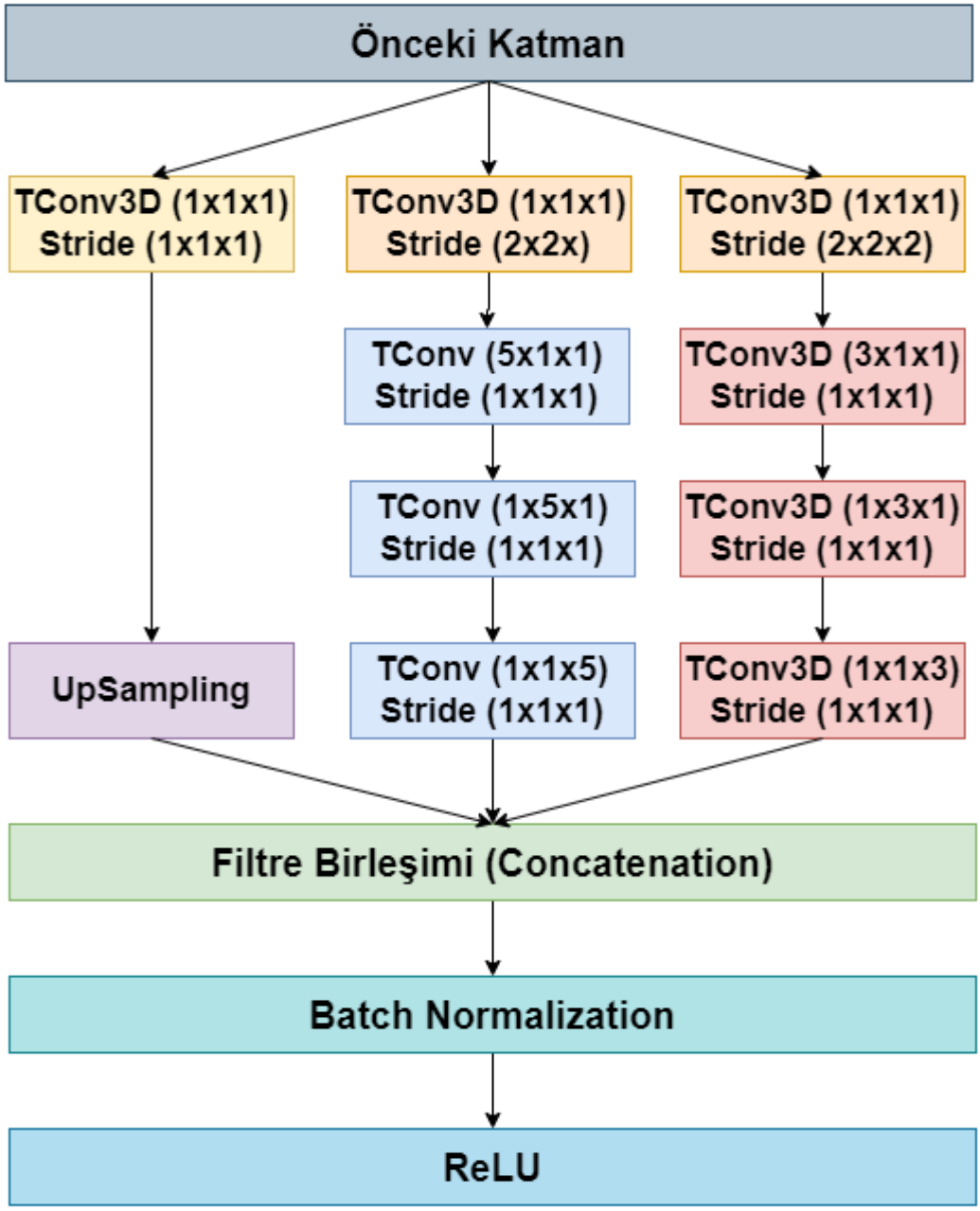
Inception ResNet 2D ağ modülü (Şekil 3.22), 2 boyutlu girdilerden atlamalı bağlantıların da kullanılması ile öznitelik çıkarmak için geliştirilmiştir. Inception ResNet 2D ağ modülü, Inception Alt Örnekleme 2D ağ modülünde olduğu gibi inception yaklaşımından faydalanılarak geliştirilmiştir. Ancak bu modülde Max Pooling katmanına yer verilmemiştir. Bu ağ modülünde, önceki katmandan gelen girdi verisi atlamalı bağlantı (skip connection) yöntemi ile modülün çıktısı verilmeden önceki çıktısı ile bağlantı kurar ve en son Relu aktivasyon fonksiyonundan geçirilerek çıktı vermektedir. Atlama bağlantısı sayesinde hem gradyan ölmesi problemi en aza indirgenilmesi hem de daha önceki katmanlardaki filtre bankaları ile elde edilmiş olan özniteliklerin tekrar hatırlatılması ve son çıktıda da etkisini göstermesi hedeflenmiştir. Bu ağ modülünde boyut arttırma veya azaltma yapılmamıştır. Bu ağ modülünde, gradyan ölmesinin azaltılması ve farklı boyutta çekirdekler (kernel) kullanılarak yapılan evrişim işlemleri ile farklı boyutlardaki özniteliklerin çıkarılması hedeflenmiştir.

3.9.4. Inception Üst Örnekleme (UpSampling) 3D Ağ Modülü

Inception Üst Örnekleme 3D ağ modülü (Şekil 3.23), 3 boyutlu girdilerin yükseklik ve genişlik eksenlerinin boyutlarının artırılması için geliştirilmiştir. Inception Üst Örnekleme 3D ağ modülü, Inception Alt Örnekleme 2D ağ modülünde olduğu gibi boyut manipülasyon işlemleri hem stride'lar ile hem de UpSampling katmanı ile sağlanmıştır. Bu modülde boyut büyütme işlemi için UpSampling katmanı ve $2 \times 2 \times 2$ stride'lı 3D devrik evrişim işlemleri kullanılmaktadır. Bu işlemlerden sonra aynı katman seviyesinde $3 \times 1 \times 1$, $1 \times 3 \times 1$, $1 \times 1 \times 3$ kernel boyutlu ve $5 \times 1 \times 1$, $1 \times 5 \times 1$, $1 \times 1 \times 5$ kernel boyutlu 3D devrik evrişim işlemi yapılmaktadır. Daha sonra yine Inception DownSampling 2D modülünde olduğu gibi filtre birleştirme, yığın normalizasyonu ve Relu katmaları kullanılmış ve çıktı üretilmiştir. Bu ağ modülünde, önceki katmandan gelen girdilerin boyutları 2 kat büyütülmüştür. Örneğin bir önceki katmandan gelen girdinin boyutu F kanal sayısı olmak üzere $16 \times 16 \times 16 \times F_1$ ise, Inception UpSampling 3D ağ modülünden sonra bu boyut $32 \times 32 \times 32 \times F_2$ olacaktır. Bu ağ modülünde, hem bir önceki katmandan gelen girdinin boyutu iki katına çıkarılması hem de iki katına çıkarılan girdilerin, farklı boyutta çekirdekler (kernel) kullanılarak yapılan evrişim işlemleri ile farklı boyutlardaki özniteliklerin çıkarılması hedeflenmiştir.



Şekil 3.22: Inception ResNet 2D Ağ Modülü



Şekil 3.23: Inception Üst Örnekleme 3D Ağ Modülü

3.9.5. Kayıp - Hata Fonksiyonu

Eğitim esnasında 3D yeniden oluşturma problemine öncelikle segmentasyon problemi gibi yaklaşılmıştır. 3D yeniden oluşturma problemine segmentation problemi gibi yaklaşılırken 3 boyutlu koordinat sisteminde (x, y, z) nesnenin bir voxel'inin bulunduğu koordinatlara 1, nesnenin bir voxel'inin bulunmadığı arkaplan (background) koordinatlarına 0 değeri atanmıştır. Bu aşamada elde edilen veri, seyreklik (sparsity) özelliği göstermektedir. Bir başka deyişle elde

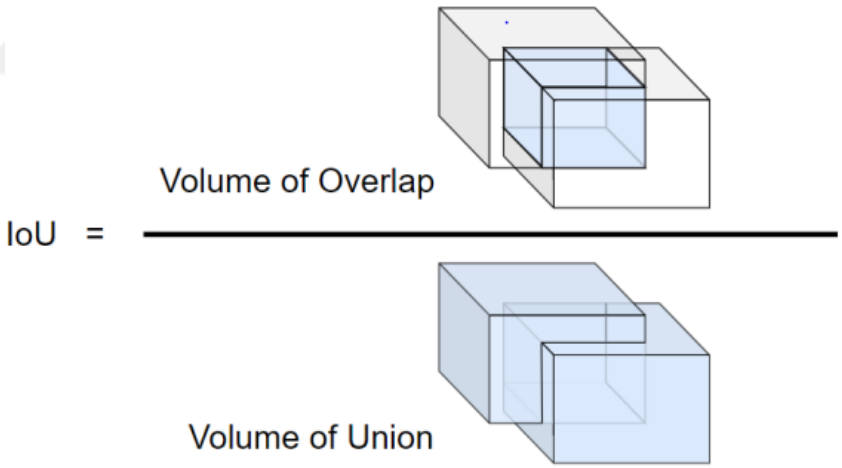
edilen veride 0 değerlerinin sayısı, 1 değerlerinin sayısından oldukça fazladır. Bu durumda ortalama kare hata, çapraz entropi gibi klasik hata fonksiyonlarında dengesiz etiket (label) verilerinde baskın olan etiket değerinin hata fonksiyonunda etkisi daha fazla olmaktadır. Hesaplanan hata değeri optimize edilirken, baskın etiketin önemi baskın olmayan etiketten daha fazla olmaktadır. Bu sorunu aşmak için Voxel verisinde dolu pikseller/koordinatlara karşılık gelen 1 ve arkaplan koordinatlarına karşılık gelen verilere 0 değeri verilmesi üzerine Denklem 3.23'de gösterildiği gibi boş pikselleri karşılık gelen n_0 adet 0 değeri için beklenen y_0 değeri ve derin öğrenme ağının çıktısı $\hat{y}_0$ değeri ile dolu piksellere karşılık gelen n_1 adet 1 değeri için beklenen y_1 değeri ve derin öğrenme ağının çıktısı $\hat{y}_1$ değerleri için ayrı ayrı kendi aralarında ortalama Cross Entropy hatası hesaplanması yapılıp en son elde edilen 2 ortalama Cross Entropy hatası toplanması ile elde edilen Kategorik Ortalama Çapraz Entropi (Categorical Mean Cross Entropy - CMCE) hata fonksiyonu geliştirilmiştir. Eğitim esnasında önerilen CMCE hata fonksiyon kullanılmıştır. Önerilen yeni hata fonksiyonu olan CMCE hata fonksiyonu ile bir voxel modelinde bulunan 0 kategorisinde veriler ile 1 kategorisindeki veriler arasındaki dengesiz dağılım problemi aşılması hedeflenmiştir.

$$CMCE(y,\hat{y}) = -\frac{\sum_{i=0}^{n_0} y_{0_i}\, log\hat{y}_{0_i}}{n_0} - \frac{\sum_{i=0}^{n_1} y_{1_i}\, log\hat{y}_{1_i}}{n_1} \tag{3.23}$$

4. BULGULAR

Bu bölümde tez çalışmasında önerilen yöntemin, Şekil 4.1'de gösterilen IoU (Intersection over Union - Kesişim bölü Birleşim) performans metriği (Denklem 4.1) kullanılarak elde edilen performans değerleri kategorik olarak ele alınmıştır. Bununla beraber önerilen Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonun, kategorik verilerin eğitiminde hata fonksiyonu olarak sıklıkla kullanılan Çapraz Entropi (CE) hata fonksiyonu arasındaki performans karşılaştırılması yine IoU performans metriği cinsinden yapılmıştır. Son olarak da örnek girdi verileri kullanılarak önerilen model ile oluşturulan Voxel Gösterimli 3 boyutlu modeller, referans modeller ile karşılaştırılması ele alınmıştır.

$$IoU = \frac{Kesişim\ (Intersection)}{Birleşim\ (Union)} \tag{4.1}$$

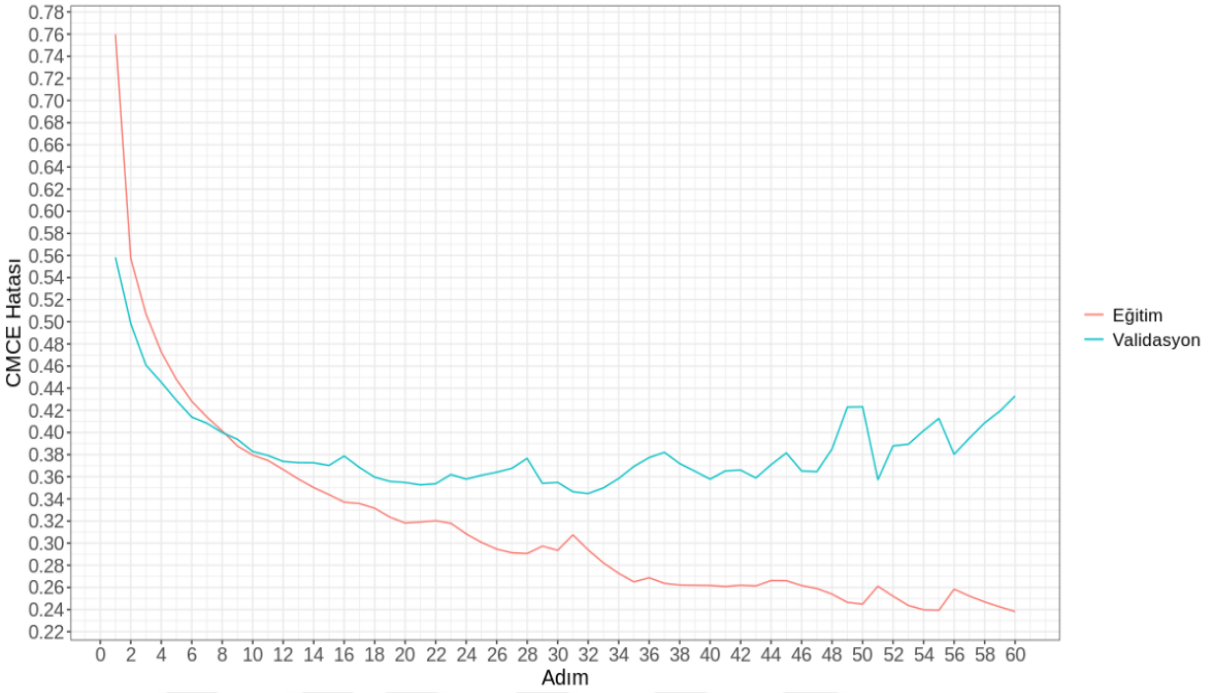


Şekil 4.1: IoU [36]

Tez çalışması sonucunda önerilen derin öğrenme modelinin Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonu ile eğitilmesi sonucu elde edilen öğrenim eğrisi Şekil 4.2'de gösterildiği gibidir. Eğitim sonucunda eğitim veri kümesi için 0.2382 ve validasyon veri kümesi için 0.4333 CMCE hatası elde edilmiştir. Ancak Şekil 4.2'de de görüldüğü üzere eğitimin validasyon veri kümesi için en başarılı olduğu adım 32. adımdır ve daha sonrasında eğitimde aşırı öğrenme (overfitting) problemi oluştuğu gözlemlenmiştir. Bu sebeple önerilen modelin

eğitimi 60. adımda sonlandırılmıştır. Bununla beraber eğitim esnasında validasyon veri kümesinin en başarılı olduğu adımı kaçırmamak için bir alt program (subroutine) geliştirilmiş ve eğitim prosedürüne dahil edilmiştir. Geliştirilen bu alt program ile validasyon veri kümesinde en az CMCE hatasına sahip olan adım tespit edilip kaydedilmektedir. Bu sayede eğitimin en verimli olduğu adım kayıt altına alınmış ve yapılan hesaplamalarda bu kayıt altına alınan adımdaki ağırlıkların (weight) olduğu model dosyası kullanılmıştır. Validasyon veri kümesinin en başarılı olduğu adım olan 32. adımda kayıt alınan modelde, eğitim veri kümesi için 0.2939 CMCE hatası ve validasyon veri kümesi için 0.3449 CMCE hatası elde edilmiştir.

Tez çalışması sonucunda elde edilen derin öğrenme modelinin performansını ölçmek için IoU performans metriği (Denklem 4.1) kullanılmıştır. IoU performans metriği referans görüntü ile elde edilen çıktının üst üste konulduğu durumda elde edilen kesişim kümesinin alanının, birleşim kümesine alanına bölünmesi ile elde edilen ve 0-1 değer aralığında bir sonuç üreten performans metriğidir. Tez çalışması sonucunda elde edilen derin öğrenme modelinin IoU performans metriği ile hesaplanan kategori bazlı IoU değerleri Tablo 4.1'de gösterildiği gibidir. Elde edilen sonuçlarda en başarılı kategori 0.7597 IoU değeri ile Araba kategorisi ve en başarısız kategori 0.3949 IoU değeri ile lamba kategorisidir. Bütün 7 kategorinin ortalama IoU değeri ise 0.5283 olarak hesaplanmıştır. Ayrıca Tablo 4.1'de önerilen derin ögrenme ağının, önerilen Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonu ile eğitilmesi ve aynı derin öğrenme modelinin Çapraz Entropi (CE) hata fonksiyonu ile aynı veriler ile eğitilmesi sonucunda elde edilen modelin başarıları IoU metriği cinsinden karşılaştırılması da kategorik olarak yapılmıştır. Yapılan karşılaştırma sonucunda 7 kategorinin 6'sında Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonu ile önerilen derin öğrenme ağının eğitimi sonucunda elde edilen model, CE hata fonksiyonu ile önerilen derin öğrenme ağının eğitimi sonucunda elde edilen modelden IoU değeri olarak daha başarılı olduğu görülmektedir. Yalnızca "silah" kategorisinde Çapraz Entropi (CE) hata fonksiyonu ile eğitilen model, Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonu ile eğitilen modelden IoU değeri olarak 0.0158 daha başarılı olduğu görülmektedir. Yine aynı iki model, bütün test verisinin IoU değerlerinin ortalaması alındığında Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonu kullanılarak eğitilen modelin, Çapraz Entropi (CE) hata fonksiyonu kullanılarak eğitilen modelin ortalama IoU değeri olarak başarılı olduğu gösterilmiştir.



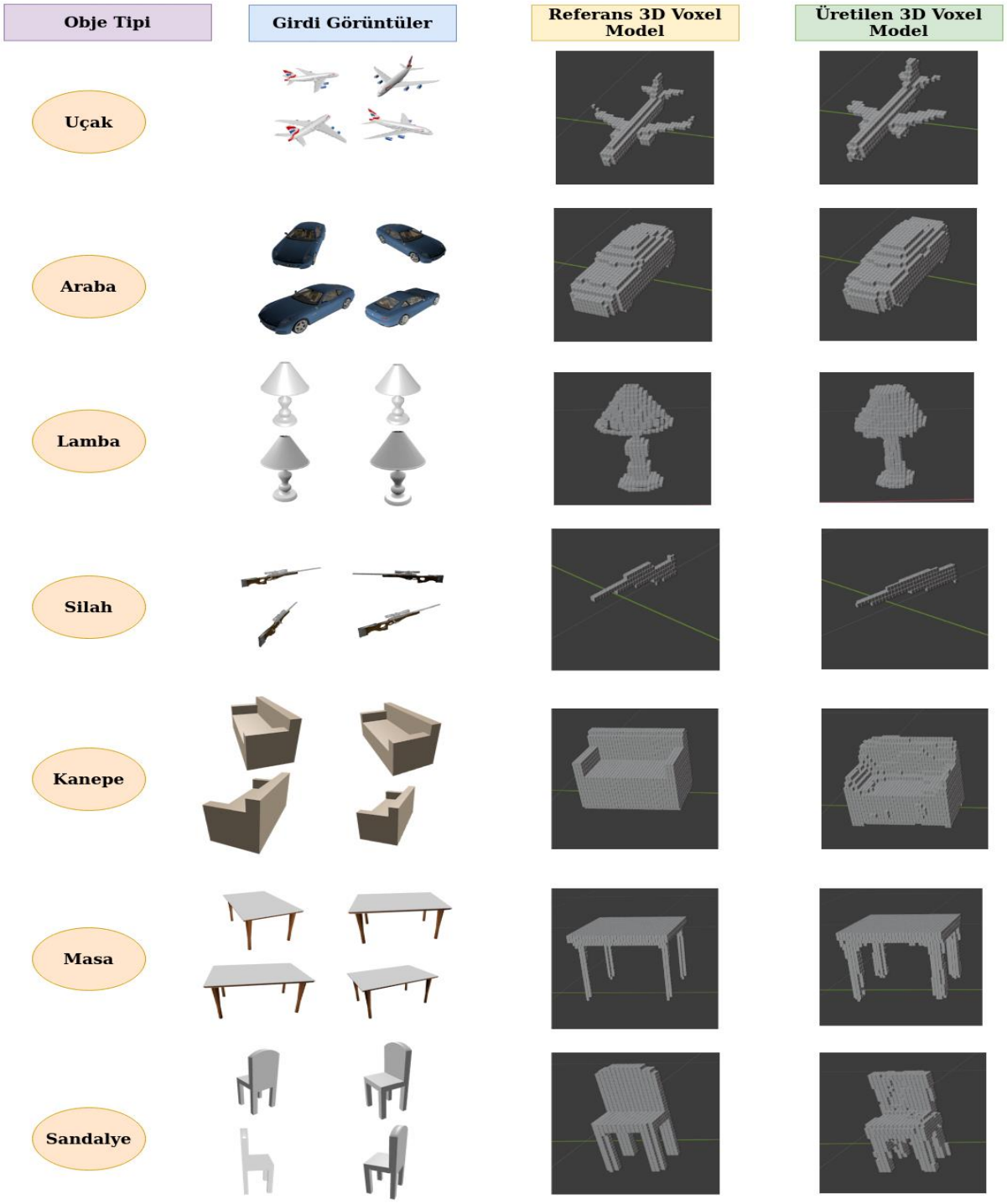
Şekil 4.2: Önerilen Modelin CMCE Hata Fonksiyonu ile Eğitiminin Öğrenme Eğrisi

Tablo 4.1: Önerilen Modelin Kategori Bazlı Kullanılan Yönteme (Hata Fonksiyonu) Göre IoU Değerlerinin Karşılaştırılması

Kategori	**IoU**	
	CMCE	**CE**
Araba	**0.7597**	0.7124
Uçak	**0.4637**	0.4383
Kanepe	**0.6011**	0.5167
Sandalye	**0.4291**	0.3953
Masa	**0.4791**	0.3979
Lamba	**0.3949**	0.3656
Silah	0.4440	**0.4598**
Ortalama	**0.5283**	0.4815

Tez çalışması kapsamında geliştirilen Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonu kullanılarak eğitilmiş olan önerilen derin öğrenme modeli sonucunda uçak, araba, lamba, silah, kanepe, masa ve sandalye kategorilerinde üretilen 3D Voxel modeller, referans

3D Voxel modelleri ile karşılaştırılması Şekil 4.3’te gösterilmektedir. Şekil 4.3’te gösterilen referans 3 boyutlu voxel modeller ile üretilen 3 boyutlu voxel modeller, bilgisayar ortamında açılmış ve ekran görüntüsü alınmıştır.



Şekil 4.3: Üretilen Örnek 3D Voxel Çıktılar

5. TARTIŞMA

Bu bölümde tez çalışmasında önerilen yöntemin, IoU performans metriği kullanılarak elde edilen performans değerleri literatürdeki diğer çalışmalar [16, 17] ile karşılaştırılması ve önerilen yöntemlerin özgün tarafları ele alınmıştır.

Tez çalışması sonucunda önerilen yöntemin hem Çapraz Entropi hata fonksiyonu ile eğitilmiş versiyonu hem de Kategorik Ortalama Çapraz Entropi hata fonksiyonu ile eğitilmiş versiyonunun kategori bazlı IoU değerlerinin, girdi sayısı 1 görüntü ve girdi sayısı 5 görüntü kullanılarak geliştirilen PTN [17] yöntemi ve McRecon [16] yöntemi ile karşılaştırılması Tablo 4.2'de gösterildiği gibidir. Tablo 4.2'de, PTN ve McRecon yöntemlerine ait IoU performans değerleri için Yan ve diğ. [17] hesapladığı değerler kullanılmıştır. Elde edilen sonuçlarda kategorisinde en başarılı yöntemin sonucu tabloda kalınlaştırılarak belirtilmiştir. Bu karşılaştırma sonucunda IoU performans metriği sonuçlarının karşılaştırılması yapılırken, ShapeNet veri kümesinin PTN yöntemi ve McRecon yönteminin eğitiminde kullanılan ortak kategorilerinin kümesi gösterilmiştir. Bu ortak kategori kümelerinin sandalye kategorisi dışındaki diğer kategorilerde önerilen yöntemin, McRecon ve PTN yöntemlerinde hem girdi görüntü sayısı 1 görüntü olan versiyonlarından hem de girdi görüntü sayısı 5 görüntü olan versiyonlarından daha yüksek IoU değerlerine ulaştığı görülmektedir. Tez çalışması sonucundaki önerilen yöntem, sadece sandalye kategorisinde en yüksek IoU değerine sahip değildir. Ancak sandalye kategorisinde de tez çalışması sonucundaki önerilen derin öğrenme ağının Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonu ile eğitilmiş versiyonu en yüksek 2. IoU değerine sahip olduğu görülmektedir. Sadece ortak kategori kümeleri baz alınıp IoU değerlerinin ortalaması alındığında ise yine önerilen derin öğrenme ağının Kategorik Ortalama Çapraz Entropi hata fonksiyonu ile eğitilmiş versiyonu diğer yöntemlerden daha yüksek ortalama IoU değerine ulaştığı Tablo 4.2'de gösterilmiştir.

Tez çalışması sonucunda önerilen derin öğrenme modelinin, tez çalışmasında ele alınan literatürdeki diğer çalışmalardan daha başarılı olmasını sağlayan unsur önerilen derin öğrenme ağında kullanılan alt modüllerdir. Derin öğrenme ağında kullanılan alt modüllerde kullanılan Inception yaklaşımında farklı boyutta çekirdekler ile evrişim işlemi yapılarak filtre bankası elde edilmiştir. Elde edilen filtre bankasında farklı çekirdek boyutlarından yakalan farklı

boyutlardaki öznitelikler, 3 boyutlu modelin oluşturulması esnasında literatürdeki diğer çalışmalardan daha ayırt edici ve anlamlı olmaktadır. Ayrıca bu öznitelikler çıkartılırken yine Inception yaklaşımı sayesinde normalden daha az parametre kullanılmaktadır. Bu sayede eğitim esnasında daha az parametrenin eğitimi yapılmıştır. Bu da 3D yeniden oluşturma problemini bir miktar daha basite indirgenmesini sağlamaktadır.

Tablo 4.2: 3D Yeniden Oluşturma Kategori Bazlı Kullanılan Yönteme Göre IoU Değerlerinin Karşılaştırması

Yöntem	**Girdi Görüntü Sayısı**	**IoU**					**Ortalama**
		Araba	Uçak	Kanepe	Sandalye	Masa	
PTN [17]	1	0.4437	0.3352	0.3309	0.2241	0.1977	0.2931
McRecon [16]	1	0.5622	0.3727	0.3791	0.3503	0.3532	0.4036
PTN [17]	5	0.6593	0.4422	0.5188	0.3736	0.3356	0.4572
McRecon [16]	5	0.6142	0.4523	0.5458	**0.4365**	0.4204	0.4849
Önerilen Model + CE	4	0.7124	0.4383	0.5167	0.3953	0.3979	0.4621
Önerilen Model + CMCE	4	**0.7597**	**0.4637**	**0.6011**	0.4291	**0.4711**	**0.5949**

Tez çalışması sonucunda elde edilen derin öğrenme modelini, ele alınan literatürdeki diğer çalışmalardan daha başarılı kılan bir diğer unsurda geliştirilen Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonudur. Bu hata fonksiyonu ile yapılan eğitim hem aynı derin öğrenme ağına göre hem de diğer çalışmalar göre daha başarılı olduğu Tablo 4.2'de gösterilmiştir. Kategorik Ortalama Çapraz Entropi (CMCE) hata fonksiyonu diğer hata fonksiyonlarına göre veri kümesindeki asimetrik veri dağılımda her veri kategorisi (1, 0 – Voxel var, Voxel yok) için kendi içerisinde normalizasyon yapmaktadır. Bu da eğitim esnasında, asimetrik veri dağılımındaki bir veri kategorisinin diğer veri kategorisine baskın olması problemini minimuma indirgemektedir.

Bu bölümde bahsedilen sebeplerden ötürü önerilen derin öğrenme ağı ve önerilen hata fonksiyonu beraber kullanıldığında, bu bölümde ele alınan literatürdeki diğer çalışmalardan daha iyi IoU performans metriği sonuçları üretmektedir.

6. SONUÇ VE ÖNERİLER

Bu bölümde yapılan tez çalışmasının ürettiği sonuçlara yer verilmektedir. Bunun yanında tez çalışmasının sonuçları, eksik ve geliştirilmesi gereken yerleri tartışılmıştır.

Yapılan bu tez çalışmasında önerilen derin öğrenme modeli ve önerilen hata fonksiyonu ile 4 farklı açıdan çekilmiş görüntülerin silüet görüntüleri kullanılarak ortalama 0.5449 IoU değeri ile voxel gösterimli 3D model üretimi yapıldığı gösterilmiştir. Bu kapsamda örnek 2D görüntüler Şekil 3.18'de gösterildiği ön işlemler yapıldıktan sonra eğitilen derin öğrenme modelinin giriş katmanına beslenmesi sonucunda üretilen voxel gösterimli 3D modeller Şekil 4.3'teki gibidir.

Üretilen 3D modeller 4 farklı açıdan çekilen görüntüler ile oluşturulmaktadır. Literatürdeki bazı çalışmalar [10, 15-17] yalnızca tek görüntü kullanarak 3D modeller üretebilmektedir ancak IoU değerleri bakımından daha düşük performans ile çalışmaktadır. Ayrıca tek görüntü kullanarak geliştirilen metotlar kategorik bakımından denetimli olarak eğitilmiştir. Bu hususta 3 veya 2 görüntü kullanarak kategori bakımından denetimsiz (unsupervised) derin öğrenme çalışmaları yapılmalıdır. Kategori bakımından denetimsiz çalışmaların geliştirilmesi, geliştirilen derin öğrenme modellerinin eğitim veri setindeki kategorilere (uçak, araba vb.) bağımlılığının azaltılması ve eğitilen derin öğrenme modelinin daha önce hiç görmediği bir objenin 3 boyutlu modelini üretebilmesini sağlayacaktır. Bu da dolaylı olarak daha az iş ve işlem gücü ile daha kapsamlı bir derin öğrenme modeli ortaya çıkartacaktır. Bu sebeple kategori bakımından denetimsiz derin öğrenme modellerinin geliştirilmesi gelecekteki çalışmalar ve sektörün ihtiyacı için önemlidir.

Bu tezde önerilen yöntem kullanılarak üretilen 3D modeller voxel gösterimlidir ancak sektörün (oyun, grafik, mimar vs.) bu tip derin öğrenme tabanlı çalışmalardan yararlanabilmesi ve daha ürünleştirilebilir çözümler sunulabilmesi için daha yumuşak (smooth) ve gerçekçi 3D modeller elde edilebilen mesh gösterimli 3D model üretimine odaklanılmalıdır. Bu konularda tez

çalışması sonucunda geliştirilen yöntem yetersiz kalmakta ve gelecek çalışmalarda geliştirilmesi gerekmektedir.

KAYNAKLAR

[1] Lee, W., Lee, B., Kim, S., Jung, H., Jeon, E., Choi, T. and You, H., 2015, 3D scan to product design: Methods, techniques, and cases, *6th International Conference on 3D Body Scanning Technologies*, 2015, 27-28

[2] Bangemann, T., Karnouskos, S., Camp, R., Carlsson, O., Riedl, M., McLeod, S., and Stluka, P., 2014, State of the art in industrial automation, *Industrial cloud-based cyber-physical systems*, 2014, 23-47

[3] Bitzidou, M., Chrysostomou, D., and Gasteratos, A., 2012, Multi-camera 3D object reconstruction for industrial automation, *International Conference on Advances in Production Management Systems*, 2012, pp. 526-533, Springer

[4] Vernon, T., and Peckham, D., 2002, The benefits of 3D modelling and animation in medical teaching, *Journal of Audiovisual media in Medicine*, 2002, 25(4), 142-148

[5] Remondino, F., Barazzetti, L., Nex, F., Scaioni, M., and Sarazzi, D., 2011, UAV photogrammetry for mapping and 3d modeling–current status and future perspectives, *International archives of the photogrammetry, remote sensing and spatial information sciences*, 2011, 38(1),

[6] Krizhevsky, A., Sutskever, I. and Hinton, G., E., 2017, ImageNet classification with deep convolutional neural networks, *Communications of the ACM*, 2017, 60 (6), 84-90

[7] Uçar, A., Demir, Y. and Güzeliş, C., 2017, Object recognition and detection with deep learning for autonomous driving applications, *SIMULATION*, 2017, 93 (9), 759-769

[8] Redmon, j., Divvala, S., Girshick, R. B., Farhadi, A., 2016, You Only Look Once: Unified, Real-Time Object Detection, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 779-788

[9] Aşıroğlu, B., Senan, S., Görgel, P., Isenkul, M. E., Ensari, T., Sezen, A., Dağtekin, M. 2022, A Deep Learning Based Object Detection System for User Interface Code Generation, *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2022,1-5

[10] Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S., 2016, 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction, *European Conference on Computer Vision (ECCV)*, 9912, 628–644

[11] Gupta, H., McCann, M. T., Donati, L., and Unser M., 2021, CryoGAN: A New Reconstruction Paradigm for Single-Particle Cryo-EM Via Deep Adversarial Learning, *IEEE Transactions on Computational Imaging*, 7, 759-774

[12] Hassner, T. and Basri, R., 2006, Example Based 3D Reconstruction from Single 2D Images, *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, 2006, 15-15

[13] Grossi, E. and Buscema, M., 2007, Introduction to artificial neural networks, *European Journal of Gastroenterology and Hepatology*, 2007, 19 (12), 1046-10541

[14] Widanagamaachchi, W. N. and Dharmaratne, A.T., 2008, 3D Face Reconstruction from 2D Images, *2008 Digital Image Computing: Techniques and Applications*, 2008, 365-371

[15] Fu, K., Peng, J., He, Q. and Zhang H., 2021, Single image 3D object reconstruction based on deep learning: A review, *Multimedia Tools and Applications*, 80 (1), 463-498

[16] Gwak, J., Choy, C. B., Chandraker, M., Garg, A. and Savarese, S, 2017, Weakly Supervised 3D Reconstruction with Adversarial Constraint, *2017 International Conference on 3D Vision (3DV)*, 2017, 263-272

[17] Yan, X., Yang, J., Yumer, E., Guo, Y. and Lee, H., 2016, Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision, *Advances in neural information processing systems*, 29

[18] Simonyan, K., Zisserman, A., 2015, Very Deep Convolutional Networks for Large-Scale Image Recognition, *2015 International Conference on Learning Representations (ICRL)*, 2015, 7-9 May, California

[19] Mosser, L., Dubrule, O., Blunt, M., 2018, Stochastic Reconstruction of an Oolitic Limestone by Generative Adversarial Networks, *Transport in Porous Media*, 2016, 125. 10.1007/s11242-018-1039-9.

[20] He, K., Zhang, X. and Ren, S., 2016, Deep Residual Learning for Image Recognition, *2016 Computer Vision and Pattern Recognition (CVPR)*, 2016, June 26th- July 1st, Las Vegas

[21] Mehrotra, R., Ansari M. A., Agrawal R., and Anand, R. S., A Transfer Learning approach for AI-based classification of brain tumors, *Machine Learning with Applications*, 2, 100003

[22] Ferguson, M., Ak, R., Lee, Y. and Law, K., 2017, Automatic localization of casting defects with convolutional neural networks, *IEEE International Conference on Big Data (Big Data)*, 2017, 1726-1735

[23] Yuniarti, A. and Suciati, N., 2019, A Review of Deep Learning Techniques for 3D Reconstruction of 2D Images, *2019 12th International Conference on Information ands Communication Technology and System (ICTS)*, 2019, 12 (7), 327-331.

[24] Riegler, G., Ulusoy, A., O. and Geiger, A., 2017, OctNet: Learning Deep 3D Representations at High Resolutions, *Computer Vision and Pattern Recognition (CVPR)*, 2017, 3577-3586

[25] Wen, C., Zhang, Y., Li, Z. and Fu, Y., 2019, Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation, *International Conference on Computer Vision (ICCV)*, 2019, 1042-1051

[26] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L. and Yu, F., 2015, ShapeNet: An Information-Rich 3D Model Repository, *arXiv* , arXiv:1512.03012

[27] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Gall, J. and Stachniss, C., 2021, Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The Semantic KITTI Dataset, *The International Journal on Robotics Research (IJRR)*, 2021, 40 (8-9), 959-967

[28] Guo, M. H., Cai, J. X., Liu, Z. N., Mu, T. J., Martin, R. R. and Hu, S. M., 2021, PCT: Point Cloud Transformer, *Computational Visual Media*, 2021, 187-199

[29] Sharkawy, A. N., 2020, Principle of neural network and its main types, *Journal of Advances in Applied & Computational Mathematics*, 2020, 7, 8-19

[30] Ioffe, S., and Szegedy, C., 2015, Batch normalization: Accelerating deep network training by reducing internal covariate shift, *In International conference on machine learning, 2015*

[31] Albawi, S., Mohammed, T. A. and Al-Zawi, S., 2017, Understanding of a convolutional neural network, *International Conferance on Engineering and Technology (ICET)*, 2017, pp. 1-6

[32] Lin, M. Chen, Q. and Yan, S., 2013, Network in network, *arXiv*, 2013, arXiv:1312.4400

[33] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015, Going deeper with convolutions, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, 1-9.

[34] Ronneberger, O., Fischer, P. and Brox, T., 2015, U-net: Convolutional networks for biomedical image segmentation, *International Conference on Medical image computing and computer-assisted intervention*, 2015, 234-241

[35] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C., 2018, A survey on deep transfer learning, *In International conference on artificial neural networks*, 2018 pp. 270-279

[36] Takahashi, M., Ji, Y., Umeda, K. and Moro, A., 2020, Expandable YOLO: 3D object detection from RGB-D images, *21st International Conference on Research and Education in Mechatronics (REM)*, 2020, pp. 1-5, IEEE.

İNTİHAL RAPORU İLK SAYFASI

Batuhan Aşıroğlu

ORİJİNALLİK RAPORU

%10	%9	%1	%6
BENZERLİK ENDEKSİ	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCİ ÖDEVLERİ

BİRİNCİL KAYNAKLAR

1	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Öğrenci Ödevi	%5
2	acikbilim.yok.gov.tr İnternet Kaynağı	%2
3	avesis.istanbulc.edu.tr İnternet Kaynağı	%1
4	Submitted to Tobb University of Economics & Technology Öğrenci Ödevi	<%1
5	journalofbigdata.springeropen.com İnternet Kaynağı	<%1
6	katalog.ticaret.edu.tr İnternet Kaynağı	<%1
7	"CRISPR/CAS9 Target Prediction with Deep Learning", 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), 2019 Yayın	<%1

KURUM İZNİ YAZILARI

Uyarı: Canlı ve cansız deneklerle yapılan tüm çalışmalar için kurum izin belgelerinin eklenmesi zorunludur. Gizlilik ve mahremiyet içeren durumlarda kurum adı kapatılmalıdır.

☐ **Kurum izni gerekmektedir.**

☒ **Kurum izni gerekmemektedir.**

Batuhan AŞIROĞLU