KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

İNSAN YÜZÜ MODİFİKASYONU İÇİN YENİ BİR GAN MODELİ

YÜKSEK LİSANS TEZİ

EMRE KARDAL

TEMMUZ 2023
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

İNSAN YÜZÜ MODİFİKASYONU İÇİN YENİ BİR GAN MODELİ

EMRE KARDAL

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"BİLGİSAYAR YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : **15/06/2023**
Tezin Savunma Tarihi : **04/07/2023**

Tez Danışmanı : **Prof. Dr. Vasıf NABİYEV**

Trabzon 2023

ÖNSÖZ

Bu araştırmada yer alan tüm/kısmi nümerik hesaplamalar TÜBİTAK ULAKBİM, Yüksek Başarım ve Grid Hesaplama Merkezi'nde (TRUBA kaynaklarında) gerçekleştirilmiştir.

EMRE KARDAL

Trabzon 2023

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “İnsan Yüzü Modifikasyonu için Yeni Bir GAN Modeli” başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Vasıf NABİYEV‘in sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 04/07/2023

EMRE KARDAL

İÇİNDEKİLER

Sayfa No

ÖZET

Yüksek Lisans Tezi

İNSAN YÜZÜ MODİFİKASYONU İÇİN YENİ BİR GAN MODELİ

EMRE KARDAL

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Vasıf NABİYEV
2023, 39 Sayfa

Günümüzde aktif olarak kullanılan Üretken Çekişmeli Ağlar (GANs) son teknoloji öğrenme algoritmalarındandır. GAN'ın özellikleri sayesinde yüksek karmaşıklıktaki örnekler işlenerek yapay görüntü, ses veya video üretilmesi mümkün hale gelmiştir. Genellikle GAN algoritması kullanan modeller rasgele gürültü örnekleri ile rasgele sonuçlar üretmektedir. Bu çalışmada, yeni bir GAN modeli tasarlandı. Model modifiye edilmesi istenen hedef yüzün, koşullara uygun sonucu oluşturulması üzerinedir. Deneylerde verilen fotoğraflar üzerinde Kadın/Erkek değişimi GAN ile kayda değer bir şekilde başarılmıştır. Aynı model Sakalsız/Sakallı değişimi içinde eğitilerek modelin genel olarak başarılı olduğu da tespit edilmiştir. Ayrıca ağın fotoğrafları eski haline döndürme durumu %90 oranına kadar başarılı olmuştur.

Anahtar Kelimeler: Üretken Çekişmeli Ağlar, Makine Öğrenmesi, Yapay Zekâ

SUMMARY

Master Thesis

NEW GAN MODEL FOR HUMAN FACE MODIFICATION

EMRE KARDAL

Karadeniz Technical University
Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Prof. Dr. Vasıf NABİYEV
2023, 39 Pages

Generative Adversarial Networks (GANs) are one of the state-of-the-art machine learning algorithms actively used today. Thanks to the capabilities of GANs, it has become possible to generate artificial images, sound or videos by processing complex examples. Models that generally use the GAN algorithm produce random results with random noise samples In this study, a new GAN model was designed. The model is about creating a result suitable for the conditions of the target face to be modified. The female/male exchange on the photographs given in the experiments was achieved with GAN in a remarkable way. The same model was trained in the Beardless/Bearded change, and it was determined that the model was generally successful. In addition, the success of the network to restore the photos to their original state was 90%.

Key Words: Generative Adversarial Networks (GANs), Machine Learning (ML), Artificial Intelligence (AI)

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

$\min_G \max_D V(D,G)$	İki oyunculu MiniMax fonksiyonu
$\tanh' t$	Tanh fonksiyonunun türevi
$\{x^{(1)}, \dots, x^{(m)}\}$	M adet gerçek veri dizisi
$\|x_i - y_i\|$	x_i değeri ile y_i değeri arasındaki mutlak fark
$\{z^{(1)}, \dots, z^{(m)}\}$	M adet rasgele gürültü dizi
∇_{θ_d}	Ayırt Edici ağın ağırlık değerlerinin gradyan vektörü
∇_{θ_g}	Üretici ağın ağırlık değerlerinin gradyan vektörü
$\log(1 - D(G(z)))$	$1 - D(G(z))$ değerinin logaritma 10 tabanında sonucu
$\log(D(x))$	$D(x)$ değerinin logaritma 10 tabanında sonucu
$\mathcal{L}_{MAE}(S, S^*)$	Mean Absolute Error fonksiyonu
$\mathcal{L}_{MSE}(S, S^*)$	Mean Squared Error fonksiyonu
$\tanh t$	Tanh fonksiyonu
$E_{x \sim p_{data}(x)}$	X için beklenen değerler dizisi
$E_{z \sim p_{data}(z)}$	Z için beklenen değerler dizisi
$ReLU'$(t)	Rectified Linear Unit aktivasyon fonksiyonunun türevi
$S'(t)$	Sigmoid/Logistic fonksiyonunun türevi
$f'(x)$	F fonksiyonunun türevi
$\sum_{i=1}^{N} x_i$	Toplam sembolü
n_t	Öğrenme oranı
α_t	Öğrenme oranı
θ_t	Theta (Ağırlığı temsil etmektedir)

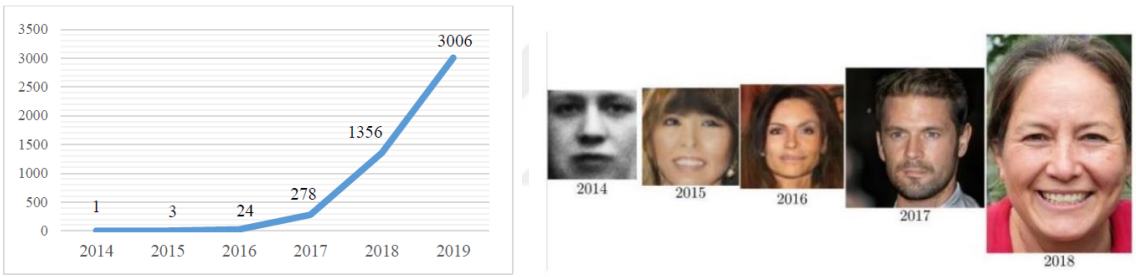
ADAM	Adaptive Moment Estimation
BCE	Binary Cross Entropy
BigGAN	Large Scale Generative Adversarial Network
CCE	Categorical Cross Entropy
CGAN	Conditional Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
GANs	Generative Adversarial Networks
LSGAN	Least Squares Generative Adversarial Network
MAE	Mean Absolute Error
MC-GAN	Multi-Conditional Generative Adversarial Network
MOMENTUM	Stochastic Gradient Descent with Momentum
MSE	Mean Squared Error
NESTEROV	Stochastic Gradient Descent with Nesterov Momentum
RMSprop	Root Mean Square Propagation
SGD	Stochastic Gradient Descent
WGAN	Wasserstein Generative Adversarial Network
$D(G(z))$	Ayırt Edicinin Üreticiden aldığı değere göre tahmin fonksiyonu
$D(x)$	Ayırt Edicinin tahmin fonksiyonu
$G(z)$	Üretici fonksiyon
$LReLU'$(t)	Leaky Rectified Linear Unit aktivasyon fonksiyonunun türevi
$LReLU$(t)	Leaky Rectified Linear Unit aktivasyon fonksiyonu
$ReLU$(t)	Rectified Linear Unit aktivasyon fonksiyonu
$S(t)$	Sigmoid/Logistic fonksiyonu
$f(x)$	F fonksiyonu

γ	İvmelenme/momentum değeri
ε	Epsilon

1. GENEL BİLGİLER

1.1. Giriş

Günümüzde güvenlik sistemleri, yüz estetiği, film endüstrisi, bilgisayar yaratıcılığı açısından yüz modifikasyonu önem taşımaktadır. Bu tarz problemlerin çözümü, yeni yöntemler ve akıllı algoritmalar gerektirmektedir. Yapay zekâ modellerinden biri olan GAN ise bu alanda kullanılabilecek yeni yöntemlerdendir. GAN, Ian Goodfellow tarafından, yarı denetimli ve denetimsiz öğrenme yöntemi olarak önerildi (Goodfellow vd., 2014). Ve gün geçtikçe daha popüler hale geldi. Şekil 1 (a)'da bu popülerliğin makale sayısına etkisi, Şekil 1 (b)'de ise GAN'ın yıllar geçtikçe gelişiminin sonucu görülmektedir.



(a) (b)

Şekil 1. GAN'ın gelişimi (Salehi vd., 2020) (a) 2014'ten 2019'a kadar GAN'lar hakkında Scopus da indekslenen makalelerin sayısı (b) 2014 yılından itibaren GAN ile yüz oluşturmadaki 4.5 yıllık ilerleme

GAN, oyun teorisindeki Nash dengesi durumunu temel alan ve birbirini eğiten iki ağdan oluşmaktadır (Goodfellow, 2016). Bu iki ağ Üretici (Generator) ve Ayırt Edici (Discriminator) olarak adlandırılır. Ayırt Edici, üretilen rasgele sonuçlar ile veri setinden gelen rasgele örneklerden hangisinin gerçek, hangisinin sahte örnekler olduğunu çözmeye çalışır. Bu ağ sonucu 0 ile 1 arasında puanlar. Bu puan ağa gelen örnekler ne kadar gerçekse 1'e yakın, ne kadar sahteyse 0'a yakın olarak değerlendirilir. Başta üretilen sonuçlar tamamen rasgele olabilir. Eğitim devam ettikçe Ayırt Edici artık daha net bir şekilde gerçek ile sahte örnekleri ayırt eder.

tamamen rasgele olabilir. Eğitim devam ettikçe Ayırt Edici artık daha net bir şekilde gerçek ile sahte örnekleri ayırt eder. Üretici ağ ise Ayırt Edicinin gerçek olduğunu sandığı örneklere yakınsamaya, sahte olduğunu anladığı örneklerden uzaklaşmaya çalışır. Bu rekabet durumu GAN'ın diğer sinir ağı modellerinden daha performanslı ve gerçekçi sonuçlar üretmesini sağlar. Bu ağların eğitim algoritması ise Algoritma 1' de gözükmektedir (Goodfellow, 2016).

Algoritma 1:

for "eğitim sayısı" do:

for "Ayırt Ediciye uygulanacak adım sayısı" do:

m adet gürültü örneği $\{z^{(1)}, \dots, z^{(m)}\}$

m adet gerçek örnekler $\{x^{(1)}, \dots, x^{(m)}\}$

Ayırt Ediciyi Artan Değişken Gradyan yöntemiyle güncelle:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

end for

m adet gürültü örneği $\{z^{(1)}, \dots, z^{(m)}\}$

Üreticiyi Azalan Değişken gradyan Yöntemiyle güncelle:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})))$$

end for

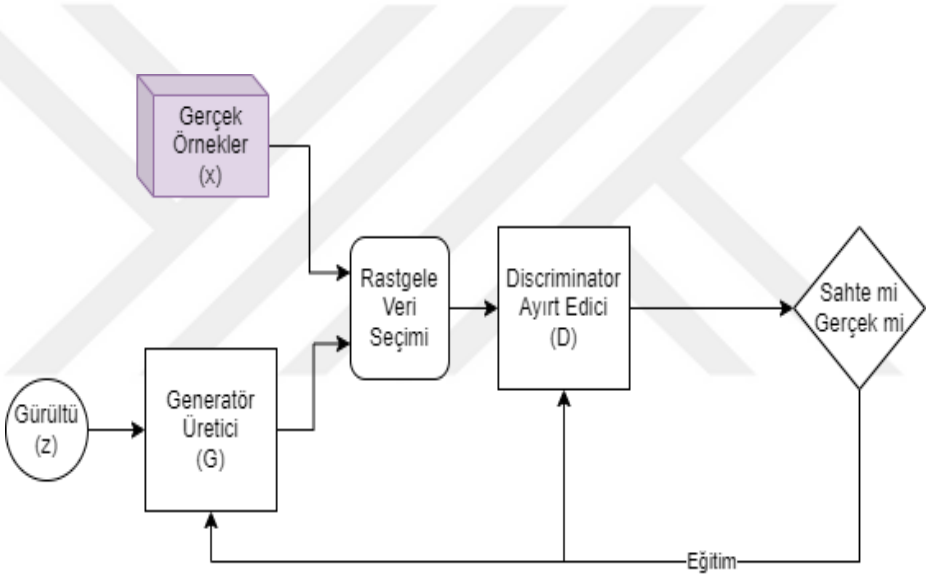
Herhangi bir gradyan tabanlı yöntem kullanılabilir (Goodfellow, 2016).

1.2. Üretken Çekişmeli Ağlar (GANs)

Orijinal GAN modeli oyun teorisine dayanmaktadır. Denklem (1)'de (Mirza & Osindero, 2014) görülen minimax fonksiyonunda, aynı anda eğitilen Üretici G olarak, Ayırt Edici ise D olarak temsil edilmektedir. Bu denklemdeki amaç en iyi sonuçları en az maliyetle üreterek eğitimin tamamlanmasıdır. Ayırt Edici D bu denklemin sonucunu maksimum hale getirmeye çalışır. Üretici ise tam tersi şekilde, denklemin sonucunu minimum hale getirmeye çalışır. Denklemdeki $D(x)$ ve $D\big(G(z)\big)$'nin alabileceği değerler "0" ila "1" arasıdır. Bu yüzden denklemin maksimum değeri $\log 1 + \log(1 - 0)$'dan "0" olmaktadır. Denklemin

minimum değeri ikinci kısımdan $\log(0) + \log(1-1)$'den $-\infty$ olmaktadır. Bu iki durum ağların rekabetinden ve kullanılan bazı aktivasyon fonksiyonlarının asla 0'a ulaşmaması gibi durumlardan dolayı asla gerçekleşmemektedir. Şekil 2'de GAN'ın eğitim şeması gösterilmektedir. İdeal durumda $\mathrm{D}(\mathrm{x}) = 0.5$ *ve* $\mathrm{D}\big(\mathrm{G}(\mathrm{z})\big) = 0.5$ olmalıdır bu durumda ise bu denklemin sonucu yaklaşık olarak $\log(0.5) + \log(1 - 0.5) = 2\log(0.5) \cong -0.602$ *'dir*. Yani tek değer için hesaplama yapıldığında bu denklemin ideal sonucu -0.602 oluştur.

$$\min_G \max_D V(D,G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_{data}(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right] \quad (1)$$



Şekil 2. Orijinal GAN akış şeması

Bu şemada GAN, rasgele verilen bir gürültüyle oluşturulan Üretici çıkışını gerçek örneklerle karıştırarak Ayırt Ediciye vermektedir. Ayırt Edici örneğin sahte veya gerçek olma durumunu değerlendirir. Sonuca göre ise geri yayılım algoritmasının (Boué, 2018) yardımıyla Ayırt Edici ve Üretici ağın katmanlarındaki ağırlıklar güncellenir. Bu algoritmada artan/azalan gradyan yöntemiyle her bir katmandaki düğümün hata üzerindeki etkisi, kısmi türevlerin yardımıyla bulunur. Bunun için kullanılan aktivasyon fonksiyonunun türevi olması gerekmektedir.

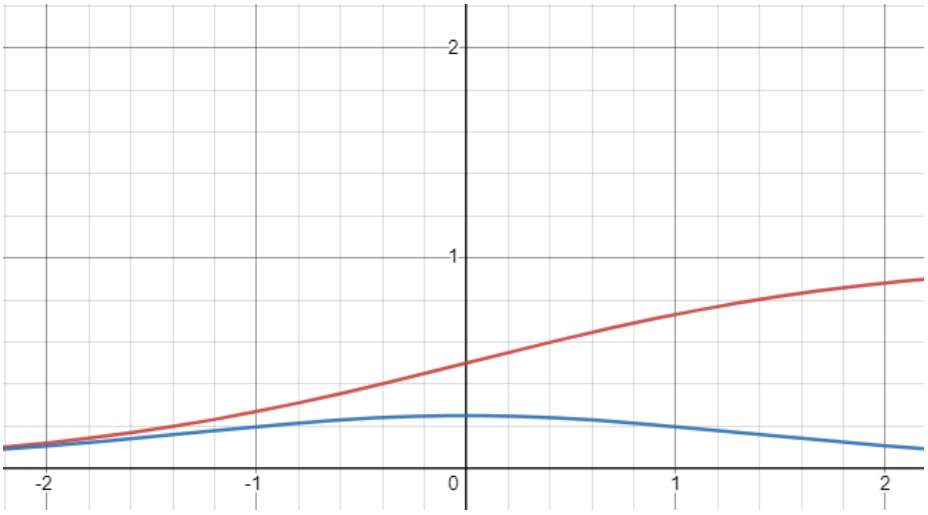
1.3. Aktivasyon Fonksiyonları

Karmaşık ve çok katmanlı eğitimlerde eğer problem lineer olarak çözülemiyorsa aktivasyon fonksiyonlarına ihtiyaç vardır (Sharma vd., 2017). Bu fonksiyonlar ayrıca önceki katmanlardan çıkan sonucun sonraki katmanlara aktarılıp aktarılmayacağı ya da ne kadar oranla aktarılacağına karar veren fonksiyonlardır (Kılıçarslan vd., 2021). Bunun yanı sıra aktivasyon fonksiyonları değerleri belirli aralıklarda sıkıştırmak istediğimizde de kullanılır (Sharma vd., 2017). Kullanılan aktivasyon fonksiyonlarından bazıları Sigmoid/Logistic, Tanh, ReLU ve LReLU'dur.

1.3.1. Sigmoid/Logistic Aktivasyon Fonksiyonu

Sigmoid aktivasyon fonksiyonu Denklem (2)'de türevi ile gösterilmektedir (Karabayır, 2018). Bu fonksiyon çıkışı denklemde de görüldüğü gibi 0 ila 1 arasında sıkıştırma yapmaktadır. Özellikle tahmin, olasılık gibi durumlarda kullanılmaktadır. GAN modellerinde Ayırt Edici verilerin gerçek veya sahte olma durumu 0 ila 1 arasında puanlandırmak için genellikle bu fonksiyonu kullanmaktadır. Tablo 1'de bu fonksiyonun ve türevinin bazı değerlerdeki sonucu gösterilmektedir.

$$S(t) = \frac{1}{1 + e^{-t}}, \quad S'(t) = S(t)\big(1 - S(t)\big) \tag{2}$$



Şekil 3. Logistic fonksiyonu ve türevinin grafiği. Kırmızı Logistic fonksiyonu, mavi ise bu fonksiyonun türevinin grafiğidir.

Tablo 1. Logistic fonksiyonu ve türevinin bazı değerlerdeki sonuçları

t	*S(t)*	*S′(t)*
-2	0.11920292202	0.10499358540
-1	0.26894142137	0.19661193324
0	0.5	0.25
1	0.73105857863	0.19661193324
2	0.88079707797	0.10499358540

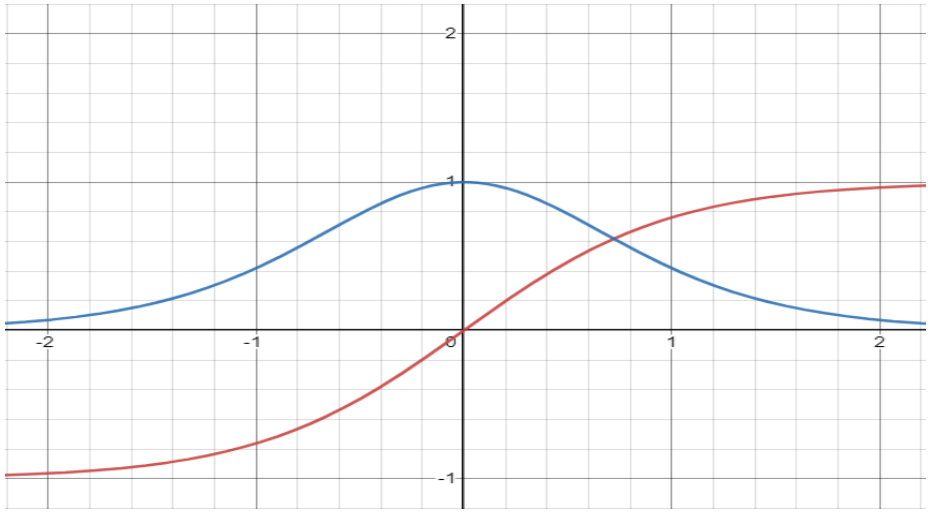
Şekil 3'te görüldüğü gibi Logistic fonksiyonu Geri Yayılım Algoritmasının uygulanması için sürekli ve türevlenebilirdir. Ayrıca bu fonksiyon monoton artan yapıdadır. +∞'da sonuç 1'e yaklaşırken -∞'da 0 olmaktadır (Turhan & Talu, 2022). Fakat bu tarz aktivasyon fonksiyonları çok katmanlı bir ağ da ara katmanlarda kullanılması durumunda kaybolan gradyanlar problemine sebep olmaktadır (Ven & Lederer, 2021; Szandała, 2021). Şekil 2'de görüldüğü gibi aşırı büyük ya da aşırı küçük değerler de Logistic fonksiyonunun türevi 0'a yaklaşmaktadır. Her katmanda Logistic fonksiyonu uygulandıkça bu değer giderek küçültmektedir. Eğitim sırasında türeve bağlı olarak her bir katmandaki ağırlıklar güncellendiği için bazı katmanlarda ve ona bağlı olan alt katmanlarda bu durum

eğitilememeye sebep olmaktadır. Bu sorundan dolayı bu tarz fonksiyonlar çok kullanılmamaktadır (Kılıçarslan vd., 2021).

1.3.2. Tanh Aktivasyon Fonksiyonu

Bu aktivasyon fonksiyonu Logistic'e benzer bir şekilde değerleri -1 ila 1 arasına sıkıştırmaktadır. Bazı GAN modellerinde Üretici tarafından üretilen verilerin bu değerler arasında olması için kullanılmaktadır. Denklem (3)'te bu aktivasyon fonksiyonu ve türevi verilmiştir (Karabayır, 2018). Şekil 4'te ise yine türeviyle birlikte fonksiyonun grafiği gösterilmektedir.

$$\tanh t = \frac{e^{t} - e^{-t}}{e^{t} + e^{-t}}, \quad \tanh' t = 1 - \tanh^2 t \tag{3}$$



Şekil 4. Tanh fonksiyonu ve türevinin grafiği. Kırmızı Tanh fonksiyonu, mavi ise bu fonksiyonun türevinin grafiğidir.

Tablo 2. Tanh fonksiyonu ve türevinin bazı değerlerdeki sonuçları

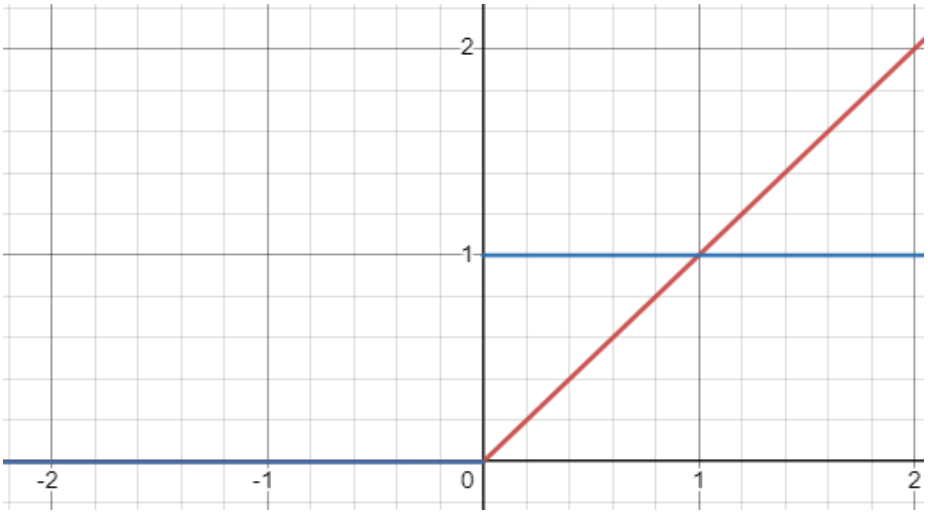
t	$Tanh(t)$	$Tanh'(t)$
-2	-0.96402758007	0.07065082485
-1	-0.76159415595	0.41997434161
0	0	1
1	0.76159415595	0.41997434161
2	0.96402758007	0.07065082485

Şekil 4'te görüldüğü gibi Tanh fonksiyonu, Logistic fonksiyonu gibi sürekli ve türevlenebilir bir yapıdadır. Logistic fonksiyonundan farklı olarak 0'a ortalanmıştır ve 0'a yaklaştıkça türevinin değeri Logistic den farklı olarak çok daha fazla artmaktadır. Bu durum 0'a yakın aralıktaki değerlerin daha fazla eğitim sırasında etkili olmasına neden olmaktadır. Logistic fonksiyonundaki gibi Tanh fonksiyonunda da kaybolan gradyanlar problemi olmaktadır (Ven & Lederer, 2021; Szandała, 2021). Bu sorunundan dolayı yine Logistic fonkiyonu gibi çok fazla kullanılmamaktadır (Kılıçarslan vd., 2021).

1.3.3. ReLU ve LReLU Aktivasyon Fonksiyonları

ReLU aktivasyon fonksiyonu Logistic ve Tanh fonksiyonlarındaki kaybolan gradyanlar problemine çözüm olarak ve daha hızlı bir eğitim için önerilmiştir (Xu vd., 2015). Denklem (4)'de bu fonksiyon gözükmektedir (Xu vd., 2015). Ayrıca Şekil 5'te ise bu fonksiyon ve türevinin grafiği ve Tablo 3'te ise bazı değerlerdeki sonuçları gösterilmektedir.

$$y = f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad , \; y' = f'(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{4}$$



Şekil 5. ReLU fonksiyonu ve türevinin grafiği. Kırmızı ReLU fonksiyonu, mavi ise bu fonksiyonun türevinin grafiği. 0'dan küçük olan kısımda ikisi sonuçta çakışık olduğu ve 0 sonucu verdiği görülmektedir.

Tablo 3. ReLU fonksiyonu ve türevinin bazı değerlerdeki sonuçları

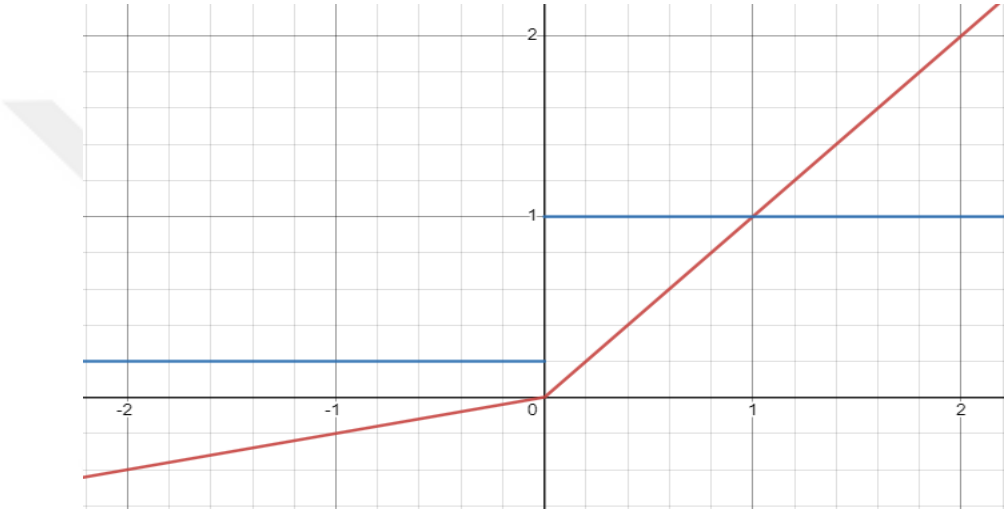
t	$ReLU(t)$	$ReLU'(t)$
-2	0	0
-1	0	0
0	0	1
1	1	1
2	2	1

ReLU aktivasyon fonksiyonu kaybolan gradyanlar problemine çözüm olsa da bir başka problem olan ölen ReLU (Lederer, 2021) problemine sebep olmaktadır. Bu problem birden fazla alt katmanın sonucu 0 olması durumunda ortaya çıkar. Bu durumda gradyanlar 0 olur ve eğitim durur. Bu durumun önüne geçmek için Sızıntılı ReLU yani LReLU kullanılabilir (Lederer, 2021). LReLU aktivasyon fonksiyonu ReLU'dan farklı olarak negatif girişlerde de bir sonuç üretmekte ve türevi de bu yüzden 0 olmamaktadır. Denklem (5)'de bu fonksiyon ve türevi (Xu vd., 2015). Şekil 6'da grafiği ve Tablo 4'te ise bazı

değerlerdeki sonucu verilmiştir. Ayrıca Şekil 7'de ReLU, LReLU, Tanh ve Logistic fonksiyonları arasındaki farkın grafiği gözükmektedir.

$$y = f(x) = \begin{cases} a.x, & x < 0 \\ x, & x \geq 0 \end{cases} \quad , \; y' = f'(x) = \begin{cases} a, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{5}$$

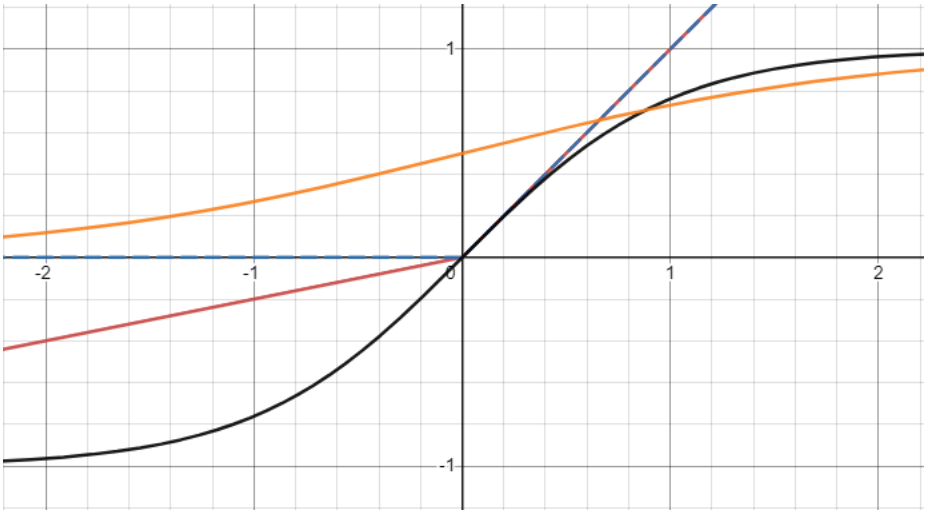
a değişkeni 0-1 arasında değer almaktadır ve pratikte genellikle 0'a yakındır.



Şekil 6. LReLU fonksiyonu ve türevinin grafiği. Kırmızı LReLU fonksiyonu, mavi ise bu fonksiyonun türevinin grafiği. a değeri bu grafik için 0.2 seçilmiştir.

Tablo 4. LReLU fonksiyonu ve türevinin bazı değerlerdeki sonuçları

t	*LReLU(t)*	*LReLU'(t)*
-2	-0.4	0.2
-1	-0.2	0.2
0	0	1
1	1	1
2	2	1



Şekil 7. Logistic, Tanh, ReLU ve LReLU fonksiyonlarının grafiği. Turuncu Logistic, siyah Tanh, mavi ReLU ve kırmızı LReLU fonksiyonu

1.4. Hata Fonksiyonları

Farklı durumlar için farklı hata fonksiyonları kullanılmaktadır. Sınıflandırma, olasılıklandırma için kullanılan bazı hata fonksiyonları Categorical Cross entropy (CCE) yani Kategorik Çapraz Entropi ve Binary Cross entropy (BCE) yani ikili Çapraz Entropi'dir. Bu tarz sınıflandırma fonksiyonlarından farklı olarak Mean Square Error (MSE) yani Ortalama Kare Hatası ve Mean Absolute Error (MAE) yani Ortalama Mutlak Hata regresyon fonksiyonları da bulunmaktadır (URL-1 ve URL-2, 2023).

1.4.1. Kategorik Çapraz Entropi ve İkili Çapraz Entropi Hata Fonksiyonları

Hata fonksiyonları hangi sınıfa ne kadar yakın olup olmadığını ölçmek için kullanılmaktadır. Kategorik Çapraz Entropi hata fonksiyonu 2 veya daha fazla sınıflı bir problem için her sınıfta olma oranının hesaplanması ve gerçekte hangi sınıfta olduğunun arasındaki farkı hesaplayarak hata oranını bulmuş olur. Denklem (6)'da Çapraz Entropi'nin formülü verilmiştir (Dorfer vd., 2015).

$$CCE = -\sum_{j=1}^{C} y_{i,j} \log\left(p_{i,j}\right) \quad (6)$$

c sınıf sayısı $y_{i,j}$ eğer sonuç o sınıfta ise 1 değilse 0, $p_{i,j}$ ise ağın verdiği sonuç 0-1 arasında (Dorfer vd., 2015).

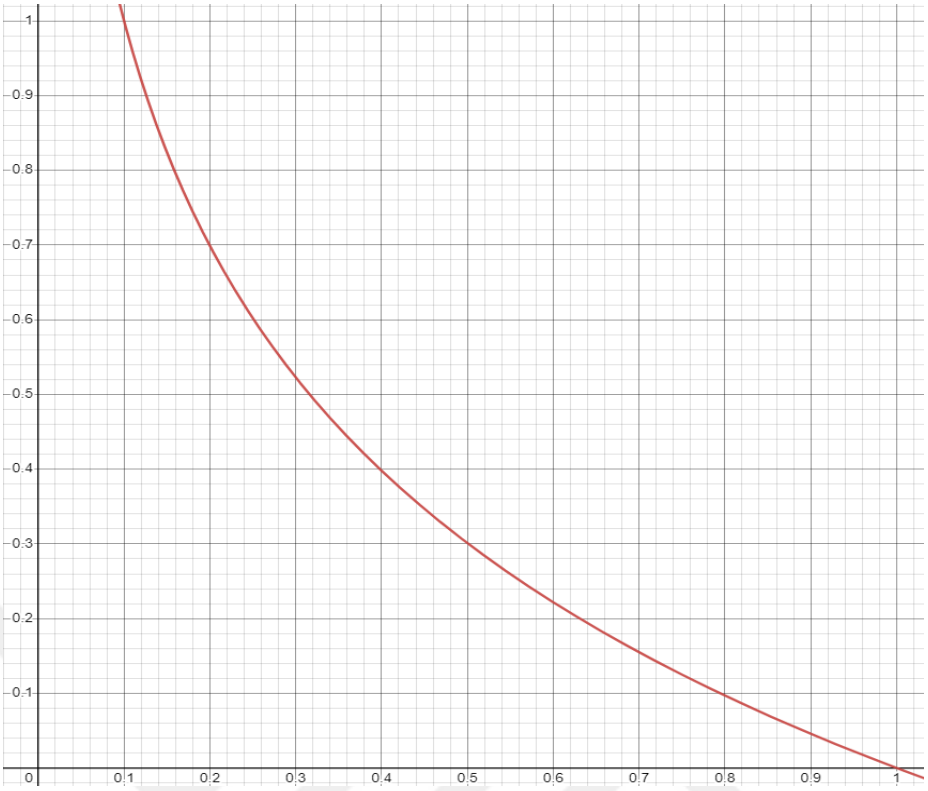
Örnek bir Kategorik Çapraz Entropi hesabı 3 sınıflı bir ağ için C=3, her sınıfta olma oranı [0.5,0.3,0.2] ve beklenen sonuç [1,0,0] yani ilk sınıfta sınıflandırılması bekleniyorsa $-(1\log(0.5) + 0\log(0.3) + 0\log(0.2)) = -\log(0.5) \cong 0.301 = \text{Hata}$

İkili Çapraz Entropi ise çoklu sınıf yerine sadece 2 sınıf için özelleştirilmiş Kategorik Çapraz Entropi hata fonksiyonudur. Bu fonksiyon tek çıkışa göre oluşturulduğundan Denklem (7)'de ki formül kullanılır.

$$BCE = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i) \quad (7)$$

Örnek bir İkili Çapraz Entropi hesabı 2 sınıflı bir ağ için C=2, 1 sonucu vermesi bekleniyorsa 2. Sınıfta, 0 sonucu vermesi bekleniyorsa ilk sınıfta olması beklenir. Bu yüzden 0.4 sonucu vermişse ve beklenen değer 0 ise ilk sınıfta olması beklenir. Denklem (8)'de ki formülün Kategorik Çapraz Entropi üzerinden türetilerek çözülmesi bu örnek için $-\sum_{j=1}^{C=2} y_{i,j} \log\left(p_{i,j}\right) => -\left((1-0)\log(1-0.4) + 0\log(0.4)\right) \cong 0.221 = \text{Hata}$ şeklinde olmaktadır.

Çapraz Entropi gibi hata fonksiyonlarında Denklem (6)'de ve Denklem (7)'de görüldüğü gibi hesaplama için negatif logaritma fonksiyonu kullanmaktadır. Negatif logaritma fonksiyonun kullanılması nedenlerinden biri Şekil 8'de görülen bu fonksiyonun grafiğinden kaynaklıdır.



Şekil 8. $-\log(x)$ fonksiyonunun 0 ila 1 arasındaki grafiği

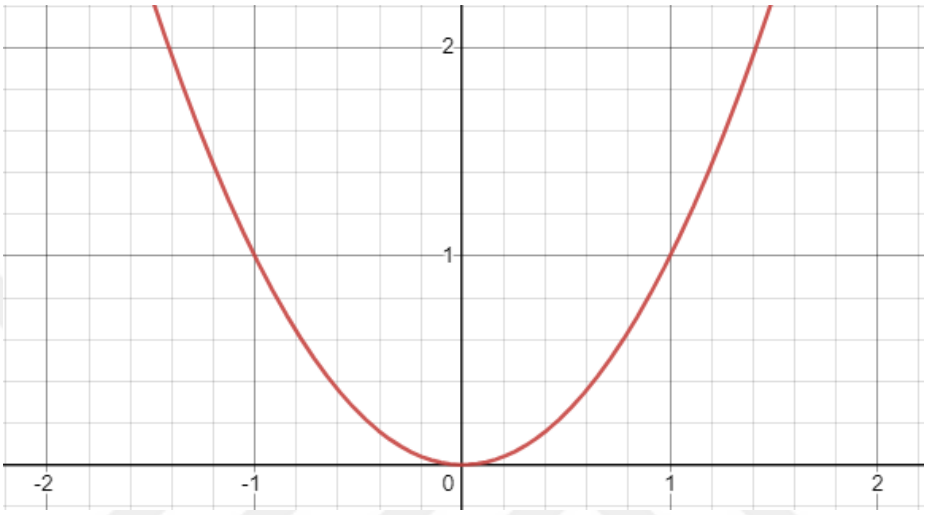
Şekil 8'de görüldüğü gibi x değeri 0'a yaklaştıkça negatif logaritma fonksiyonunun değeri katlanarak artmaktadır. Bu da tahmin sonucu, yanlış tahmin oranı fazla ise hata oranının etkisini dahada yükseltmektedir.

1.4.2. Ortalama Kare Hatası ve Ortalama Mutlak Hata Fonksiyonları

Ortalama Kare Hata fonksiyonu iki değer dizisinin arasındaki farkın karesinin ortalamasını hesaplayarak hata oranını vermiş olur. Denklem (8)' de bu fonksiyon gözükmektedir (Qi vd., 2020). Bu denklemin tek boyutlu bir örnek için bazı değerlerdeki grafiği ise Şekil 9'da görülmektedir.

$$\mathcal{L}_{MSE}(S, S^*) = \frac{1}{N}\sum_{i=1}^{N}\|x_i - y_i\|^2 \quad (8)$$

MSE, N adet tahmin bulunduran $S = \{x1, x2, \ldots, xN\}$ vektörü ile gerçekte olması gereken değerleri bulunduran $S^* = \{y1, y2, \ldots, yN\}$ vektörü arasındaki ortalama büyüklüğe karşılık gelen kaybı ölçen ikinci dereceden bir puanlama kuralıdır (Qi vd., 2020).

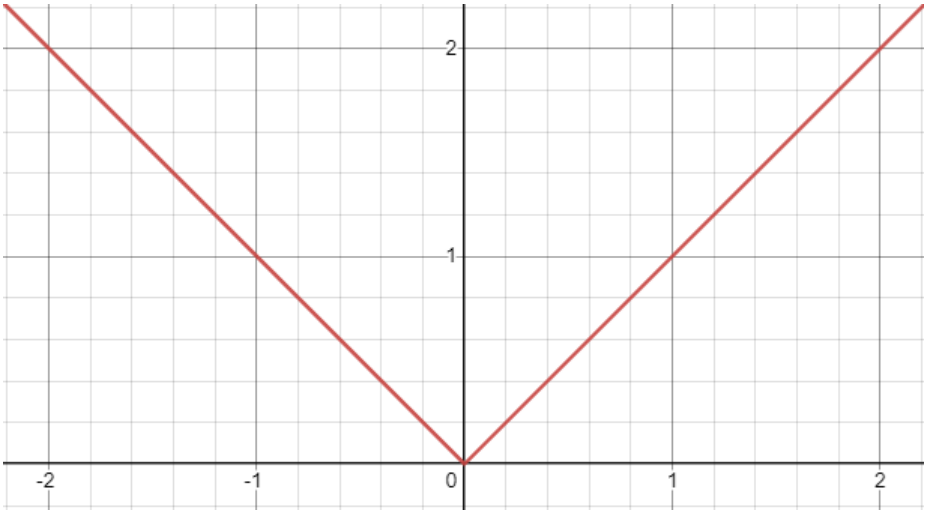


Şekil 9. Ortalama Kare Hata fonksiyonunun grafiği. y_i "beklenen değeri" bu grafik için 0 olarak seçildi.

Ortalama Mutlak Hata fonksiyonu, Ortalama Kare Hata fonksiyonuna benzer olarak iki değer arasındaki farkı alır ama bu fonksiyon karesini almak yerine mutlak değerini alıp tüm değerlere ortalamasıyla hata oranını bulur. Denklem (9)'da bu fonksiyon gözükmektedir. Ayrıca Şekil 10'da tek boyutlu bir örnek için bazı değerlerdeki grafiği gösterilmektedir.

$$\mathcal{L}_{MAE}(S, S^*) = \frac{1}{N}\sum_{i=1}^{N}\|x_i - y_i\| \quad (9)$$

MAE, N adet tahmin bulunduran $S = \{x1, x2, \ldots, xN\}$ vektörü ile gerçekte olması gereken değerleri bulunduran $S^* = \{y1, y2, \ldots, yN\}$ vektörü arasındaki ortalama büyüklüğe karşılık gelen kaybı ölçer (Qi vd., 2020).

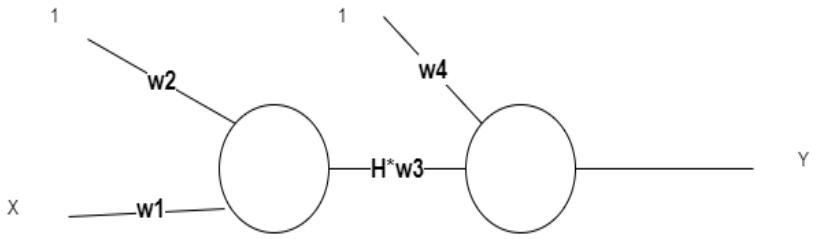


Şekil 10. Ortalama Mutlak Hata fonksiyonunun grafiği. y_i "beklenen değeri" 0 için.

1.5. Geri Yayılım Algoritması

Geri Yayılım Algoritması çok katmanlı yapay sinir ağlarında kullanılan bir yöntemdir. Bu yöntem ileri beslemeli yapay sinir ağının geriye doğru hatalarının oranını bulmamızı sağlar. Bu algoritma bir denetimli öğrenmeyi kullanmaktadır (Gershenson, 2003). Girişten gelen veriler, her bir ağırlıklarla çarpılarak ve aktivasyon fonksiyonlarından geçerek bir sonraki katmana geçer. Aradaki gizli katman sayısı bir veya daha fazla olabilir. En son çıkış katmanında bir sonuç alırız. Beklenen sonuç ile alınan sonuç arasındaki fark hata fonksiyonları ile hesaplanır. Geri Yayılım Algoritmasında amaç bu hatayı en aza indirerek istenen sonuca ulaşmaktır (Gershenson, 2003). Bir giriş vererek her katmanda türevlenebilir aktivasyon fonksiyonlarından geçerek ve en son çıkan sonuç ile istenen/beklenen sonuç arasındaki farkın hata oranını herhangi bir hata fonksiyonu ile ölçeriz. Sonra artık Geri Yayılım Algoritmasını kullanarak hatanın girdilere, ağırlıklara nasıl bağlı olduğunu bulunabilinir. Geri Yayılım Algoritması bunu türevin zincir kuralı kullanarak yapar. Böylece her ağırlığın gradyan vektörünü bulmuş oluruz. Bu adımlardan sonra artık gradyan iniş yöntemini kullanarak her bir ağırlık güncellenerek hata azaltılabilinir. Gradyan iniş yöntemleri olarak optimizasyon yöntemi algoritmaları tarafından farklı yöntemlerle kullanılır. Aşağıda Geri Yayılım Algoritması kullanılarak yapılan bir örnek gösterilmektedir. Örneğin basitleştirilmesi amacıyla aktivasyon fonksiyonu LReLU aktivasyon fonksiyon, Hata fonksiyonu ise Ortalama Mutlak Hata fonksiyonu seçilmiştir. Öğrenme oranı 0.1 ve

optimizasyon yöntemi algoritması olarak Azalan Değişken Gradyan yöntemi kullanılmıştır. Örnek ağın şekli ise Şekil 11'de gözükmektedir.



Şekil 11. Basit bir sinir ağı görseli. Ağın denklemi H = LReLU(w1*X + w2*1), Y = w3*H + w4*1, Y sonucu, X ise girişi göstermektedir.

w1 = 0.5, w2 = 1, w3 =0.5, w4 = 1 rasgele başlangıç değerleri ve Y = X^2 beklenen sonucu için X = 2'e göre hata ölçümü ve güncelleme adımları sırasıyla:

1) 0.5*2 + 1*1 = 2
2) LReLU(2) = 2 = H
3) 0.5*2 + 1*1 = 2 = Y
4) Beklenen değer X^2 ise 2^2= 4
5) Sonuç Y = 2
6) MAE = $|sonuç - Beklenen\ değer|$ = $|2 - 4|$ = 2
7) Kısmı türevin zincir kuralı uygulanır hatanın w4 üzerindeki etkisi $\frac{\partial E}{\partial w4} = \frac{\partial E}{\partial Y} * \frac{\partial Y}{\partial w4}$
8) $\frac{\partial E}{\partial Y}$ Hatanın Y'e göre türevi demek olup 6. Adımdaki MAE fonksiyonunun Y'e göre türevi -1'dir. $\frac{\partial E}{\partial w4} = -1 * \frac{\partial Y}{\partial w4}$, $\frac{\partial Y}{\partial w4}$ ise Y = (H*w3 * 1*w4) Y'nin w4'e göre türevi 1 olduğundan $\frac{\partial E}{\partial w4} = -1$ olur.
9) $\frac{\partial E}{\partial w3} = \frac{\partial E}{\partial Y} * \frac{\partial Y}{\partial w3} = -1 * \frac{\partial Y}{\partial w3}$ ise Y'nin w3 e göre türevi (H*w3 * 1*w4)' den H olur. H = 2 olduğundan $\frac{\partial E}{\partial w3} = -2$ olur.
10) $\frac{\partial E}{\partial w2} = \frac{\partial E}{\partial Y} * \frac{\partial Y}{\partial H} * \frac{\partial H}{\partial w2}$'den hesaplanır. $\frac{\partial E}{\partial Y} = -1, \frac{\partial Y}{\partial H}$ ise Y = (H*w3 * 1*w4) H'ye göre türevinden w3 yani w3= 0.5 olduğundan, $\frac{\partial E}{\partial w2} = -1 * 0.5 * \frac{\partial H}{\partial w2}$ $H =$

$\mathrm{LReLU}(w1 * X + w2 * 1)$ olduğundan $w1 * X + w2 * 1 > 0$ ise (*Bu örnek için büyük*) $\frac{\partial H}{\partial w2} = w1 * X + w2 * 1$ denklemin de H'nin w2'e göre türevi 1'dir. Bu durumda $\frac{\partial E}{\partial w2} = -1 * 0.5 * 1 = -0.5$

11) $\frac{\partial E}{\partial w1} = \frac{\partial E}{\partial Y} * \frac{\partial Y}{\partial H} * \frac{\partial H}{\partial w1}$'den hesaplanır. $\frac{\partial E}{\partial Y} = -1$, $\frac{\partial Y}{\partial H} = 0.5$ olduğuna ve w2'nin kısmi türevinde de gördüğümüz gibi bu örnek için $w1 * X + w2 * 1 > 0$'dan büyük olduğundan $\frac{\partial H}{\partial w1} = X$ yani 2 (*Başlangıçta X = 2 seçilmişti*). Bu durumda $\frac{\partial E}{\partial w1} = -1 * 0.5 * 2 = -1$
12) Azalan Değişken Gradyan yöntemine göre yeni ağırlıklar w = w – 0.1 * $\frac{\partial E}{\partial w}$ olduğundan w4 = w4 – 0.1 * $\frac{\partial E}{\partial w4}$, w3 = w3 – 0.1 * $\frac{\partial E}{\partial w3}$, w2 = w2 – 0.1 * $\frac{\partial E}{\partial w2}$, w1 = w1 – 0.1 * $\frac{\partial E}{\partial w1}$
13) $w4 = 1 - 0.1 * (-1) = 1.1, w3 = 0.5 - 0.1 * (-2) = 0.7, w2 = 1 - 0.1 * (-0.5) = 1.05, w1 = 0.5 - 0.1 * (-1) = 0.6$
14) 0.6*2 + 1.05*1 = 2.25
15) LReLU(2.25) = 2.25 = H
16) 0.7 * 2.25 + 1.1*1 = 2.675
17) Beklenen değer $2^2 = 4$
18) Sonuç Y= 2.675
19) MAE = $|sonuç - Beklenen\ değer|$ = $|2.675 - 4|$ = 1.325
20) İşlemleri 1 Adım daha tekrar ediyoruz
21) $\frac{\partial E}{\partial w4} = -1 * 1 = -1$
22) $\frac{\partial E}{\partial w3} = -1 * 2.25 = -2.25$
23) $\frac{\partial E}{\partial w2} = -1 * 0.7 * 1 = -0.7$
24) $\frac{\partial E}{\partial w1} = -1 * 0.7 * 2 = -1.4$
25) $w4 = 1.1 - 0.1 * (-1) = 1.2, w3 = 0.7 - 0.1 * (-2.25) = 0.925, w2 = 1.05 - 0.1 * (-0.7) = 1.12, w1 = 0.6 - 0.1 * (-1.4) = 0.74$
26) 0.74*2 + 1.12*1 = 2.6
27) LReLU(2.6) = 2.6 = H
28) 0.925*2.6 + 1.2*1 = 3.605
29) MAE = $|3.605 - 4|$ = 0.395

Örnekte de görüldüğü gibi hata azalmaktadır. Hata istenen orana azalana kadar farklı sayılar ile bu adımlar tekrar edilir. Sadece tek sayı yerine birden fazla farklı sayılarla yapılan işlemlerde genel ağırlık değerleri elde edilebilir. Böylece ağ Y=X^2 denklemine daha fazla benzer.

1.6. Optimizasyon Yöntemi Algoritmaları

Optimizasyon yöntemi algoritmaları ağırlıkları güncellerken kullandığımız algoritmalardır. Bu algoritmalar hata oranını daha hızlı azaltmak, yerel minimum gibi değerlere takılmamak gibi özelliklere sahiptir. Algoritma ağırlığın gradyan vektörüne ihtiyaç duyar. Bu yüzden geri yayılım algoritması ile kısmi türevin zincir kuralını kullanarak o ağırlıktaki hatanın gradyan vektörünü buluruz. Optimizasyon yöntemi algoritmasının temeli Azalan Değişken Gradyan yöntemi olup denklemi ise Denklem (10)'da ki gibidir (Choi vd., 2019).

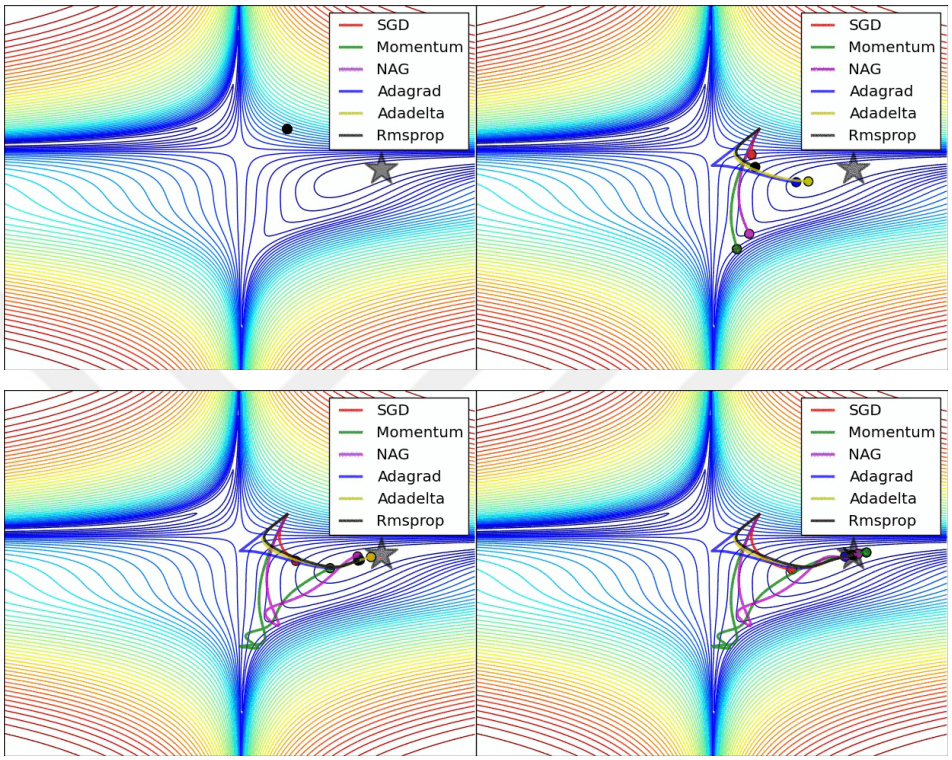
$$\theta_{t+1} = \theta_t - n_t \nabla \ell(\theta_t) \tag{10}$$

$\nabla \ell(\theta_t)$ burada θ_t ağırlığındaki gradyan vektörünü temsil etmektedir. n_t ise öğrenme oranını, θ_t ağırlığın şu an ki değeri θ_{t+1} ise ağırlığın güncellenen değeri temsil etmektedir. Optimizasyon yöntemi algoritmalardan bazıları Azalan Değişken Gradyan (SGD), Momentumlu Azalan Değişken Gradyan (MOMENTUM), Nesterov Momentumlu Azalan Değişken Gradyan (NAG), Kök Ortalama Kare Yayılımı (RMSprop), Uyarlanabilir Moment Tahmini (Adam) gibi bazı optimizasyon yöntemi algoritmaları bulunmaktadır.

1.6.1. Azalan Değişken Gradyan (SGD) Algoritması

Azalan Değişken Gradyan algoritması en temel ağırlık güncelleme algoritmasıdır. Bu algoritmanın çalışma yöntemi Denklem (10)'da ki gibidir. Denklemde görüldüğü gibi bu

algoritma sabit bir öğrenme oranına sahiptir. Her adım bu sabit öğrenme oranına bağlı olarak güncellenmektedir. Bazı optimizasyon yöntemi algoritmalarının karşılaştırılması Şekil 12'de gösterildiği gibidir (URL-23, 2023).



Şekil 12. Optimizasyon yöntemi algoritmalarının karşılaştırılması (URL-3, 2023)

Şekil 12'de görüldüğü gibi Azalan Değişken Gradyan algoritması bu karşılaştırmada hatayı en son minimize eden yöntem olmaktadır. Bu durum minimum sonuca ulaşmak için daha fazla eğitime ihtiyaç olmasına sebep olmaktadır.

1.6.2. Momentumlu Azalan Değişken Gradyan (MOMENTUM) Algoritması

Bu algoritma standart Azalan Değişken Gradyan algoritmasına ekstra bir momentum değişkeni eklenerek ivmelendirilmesini ve kaybın daha hızlı azalmasını sağlamaktadır. Ayrıca standart algoritma ani kavisli sonuçlarda yerel minimum gibi durumlarda takılı

kalmaktadır (Ruder, 2016). Denklem (11)'de bu fonksiyon gösterilmektedir (Choi vd., 2019). Denklemden de görüldüğü gibi standart algoritmadan sadece ufak bir farkı bulunmaktadır. Fakat bu fark bile Şekil 12'de görüldüğü gibi eğitime büyük etki yapabilmektedir.

$$v_0 = 0, v_{t+1} = \gamma v_t + \nabla \ell(\theta_t), \theta_{t+1} = \theta_t - n_t v_{t+1} \quad (11)$$

γ değeri 0'dan ∞'a kadar bir değer seçilebilir. Her adımda ağırlık değerinin yanı sıra v_t değeri de güncellenerek sonraki ağırlık güncellemelerinde ivmeli bir şekilde hata daha hızlı minimize sonuca ulaşır.

1.6.3. Nesterov Momentumlu Azalan Değişken Gradyan (NAG) Algoritması

Nesterov yöntemi standart Momentumlu Azalan Değişken Gradyan algoritmasından biraz farklı olarak v_t değerinin güncellemesinde kullanılan hatanın o ağırlık için olan gradyan vektörünün yanı sıra doğrudan ağırlığın güncellenmesinde de bu gradyan vektör kullanılmaktadır. Bu algoritmada sadece ivmelenmek yerine hataya bağlı olarak daha akıllıca ivmelenmesi sağlamaktadır (Ruder, 2016). Fonksiyonu Denklem (12)'de gözükmektedir (Choi vd., 2019).

$$v_0 = 0, v_{t+1} = \gamma v_t + \nabla \ell(\theta_t), \theta_{t+1} = \theta_t - n_t(v_{t+1} + \nabla \ell(\theta_t)) \quad (12)$$

1.6.4. Kök Ortalama Kare Yayılımı (RMSprop) Algoritması

Kök Ortalama Kare Yayılımı algoritması Geoff Hinton (2020) tarafından Coursera Sınıfı Ders 6e'de önerilen yayınlanmamış, uyarlanabilir bir öğrenme hızı yöntemidir (Ruder, 2016; Hinton vd., 2020). Bu yöntemin formülü Denklem (13)'de gösterildiği gibidir (Choi vd., 2019).

$$v_0 = 1,\ m_0 = 0, v_{t+1} = pv_t + (1-p)\nabla\ell(\theta_t)^2,\ m_{t+1} = \gamma m_t + \frac{n_t}{\sqrt{v_{t+1}+\varepsilon}}\nabla\ell(\theta_t)$$

$$\theta_{t+1} = \theta_t - m_{t+1} \quad (13)$$

Şekil 12'de verilen örnek için yine bu algoritmanın eğitim de ki istenilen sonuca ulaşma hızı görsel olarak verilmiştir. ε değeri sıfıra bölünmeyi önlemek ya da v_{t+1} değerinin sıfıra oldukça yakın olması durumu için eklenmiştir.

1.6.5. Uyarlanabilir Moment Tahmini (Adam) Algoritması

Adam algoritması iki algoritmanın avantajlarını birleştirilmesiyle oluşturulmuş bir optimizasyon yöntemi algoritmalarıdır. Bu yöntemlerden seyrek gradyanlarda iyi çalışan AdaGrad ile sabit ve sabit olmayan ortamlarda iyi çalışan RMSprop algoritmasıdır (Kingma & Ba, 2014). Bu algoritmanın formülü Denklem (14)'de gösterildiği gibidir (Choi vd., 2019).

$$v_0 = 0,\ m_0 = 0, m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla\ell(\theta_t), v_{t+1} = \beta_2 v_t + (1-\beta_2)\nabla\ell(\theta_t)^2$$

$$b_{t+1} = \frac{\sqrt{1-\beta_2^{t+1}}}{1-\beta_1^{t+1}},\ \theta_{t+1} = \theta_t - \alpha_t \frac{m_{t+1}}{\sqrt{v_{t+1}}+\varepsilon} b_{t+1} \quad (14)$$

Buradaki ε değeri Kök Ortalama Kare Yayılım (RMSprop) algoritmasındaki gibi sıfıra bölünmeyi ya da sıfıra oldukça yakın bir v_{t+1} değeri olması durumuna karşı eklenmiştir.

2. LİTERATÜR ARAŞTIRMASI

2.1. GAN Modelleri

GAN'ın birçok farklı modeli bulunmaktadır. Her model problem veya amaca göre değişiklik göstermektedir. Çıkan sonuçların daha iyi olması üzerine çalışan bazı modeller bulunmaktadır. Bunlardan biri ağ yapısında Convulution (Evrişim) ve DeConvulution (Ters Evrişim) kullanan bir modele sahip olan DCGAN'dır (Radford vd., 2015). Bu model özellikle resimler, fotoğraflar oluşturma üzerinedir. Ayırt Edici evrişimsel ağ yapısına sahip olup aldığı görüntüyü evrişimsel katmanlardan ve filtrelerden geçirerek bir değer döndürür. Üretici ise ters evrişimsel ağ yapısını kullanarak gelen rasgele gürültü dizisini filtreler ile katmanlardan geçirerek çıkışa bir görüntü olarak verir. Bu ağa benzer bir diğer model BigGAN'dır (Brock vd., 2018). Bu model kademeli olarak görüntünün çözünürlüğünü arttırarak daha dengeli bir yapı elde etmek üzerinedir. Katmanlar halinde olan Üretici ağ, her katmanda bir önceki katmanın çıkışını alır ve sonra görüntünün boyutunu iki katına çıkarıp bir sonraki katmana verir. Ayırt Edici ise bunun tam tersi olarak her katmanda görüntüyü küçültür. Bunu yaparken her katmanda özellik çıkarır ve son karar katmanında bunları kullanarak bir sonuca varır. Bu sonuç orijinal GAN modelindeki gibi 0 ila 1 arasında bir sayıdır.

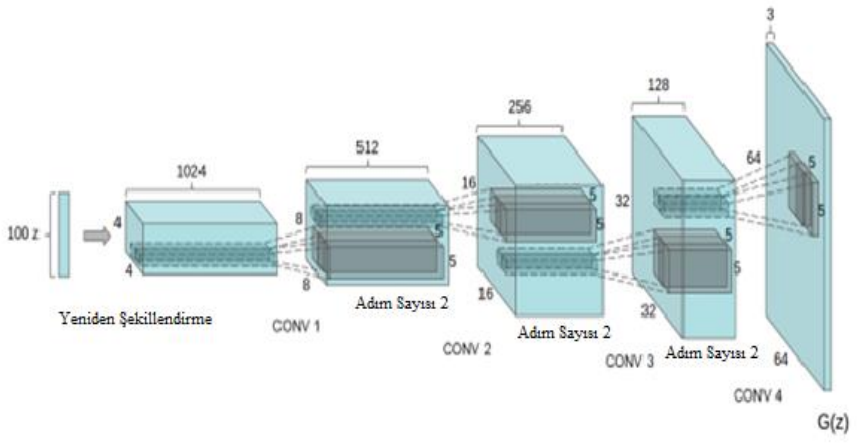
Eğer problem ayrıca koşullu sonuçlar üretilmesini gerektiriyorsa başka bir GAN modelinden faydalanılmalıdır. CGAN (Mirza & Osindero, 2014) bu modellerden biridir. CGAN diğer GAN modellerinden farklı olarak öğrenme sonucunda rasgele sonuçlar üretmektense belli bir kural çerçevesinde sonuçlar üretilmesi üzerinedir. Orijinal GAN modelinde, Ayırt Edici ağa sahte veya gerçek örnekler verilir ve bunları ayırt etmesi istenir. Üretici ise rasgele oluşturulmuş gürültü dizisiyle ürettiği sahte örneklerin, ne kadarının Ayırt Edici tarafından tespit edildiğine göre kendini günceller. CGAN'da ise Üretici ve Ayırt Edicinin girişlerine ekstradan belirlenen kurala göre oluşturulan bir veri eklenir. Böylece Ayırt Edici oluşturulan örnekleri, sahte olup olmadığının yanı sıra belirtilen kurala da uyup uymadığını kontrol eder. Ayırt Edicinin bu özelliği sayesinde Üretici gerçekçi örnekler üretse de belirtilen kurala uymadığında Ayırt Edici tarafından düzenlenmesi için zorlanır. Buradaki

kurallar örneklerin türleri modele ve istenen sonuca göre değiştirilebilinir. Aynı amaçla MC-GAN (Park vd., 2018) ise tek bir kural yerine çok koşullu kurallar ile sonuçlar üretilmesi içindir.

GAN'da oluşabilen problemlere karşı ise farklı GAN modeli önerilmiştir. Bunlardan biri WGAN'dır (Arjovsky vd., 2017). Bu modelde orijinal GAN modelindeki kayıp/hata fonksiyonu olan Jensen-Shannon Uyuşmazlığı yerine Wasserstein Mesafesi kullanılır. Bu sayede bazı durumlarda oluşan mod çökmesi, gradyanların kaybolması gibi eğitimi etkileyen etkenler minimize edilmiş olur. WGAN'a benzer olarak yine aynı sorunların önüne geçmek için LSGAN (Mao vd., 2017) modelinde en küçük kareler hata fonksiyonunun kullanılmaktadır.

2.1.1. DCGAN

DCGAN, resim/fotoğraf gibi veriler oluşturmak üzere evrişimsel katmanlar ve filtreler kullanır. Bu katmanlar ve filtreler veriler üzerinden bir takım özellik matrisi çıkarır. Her katmanda giriş bir önceki katmanın çıkarttığı özellik matrislerini de kullanarak görüntüyü oluşturmaya ya da ayrıştırmaya çalışır. Şekil 13'te (Radford vd., 2015) DCGAN modelinin Üretici (G) ağının nasıl katman ve adımlardan geçtiği gösterilmiştir.

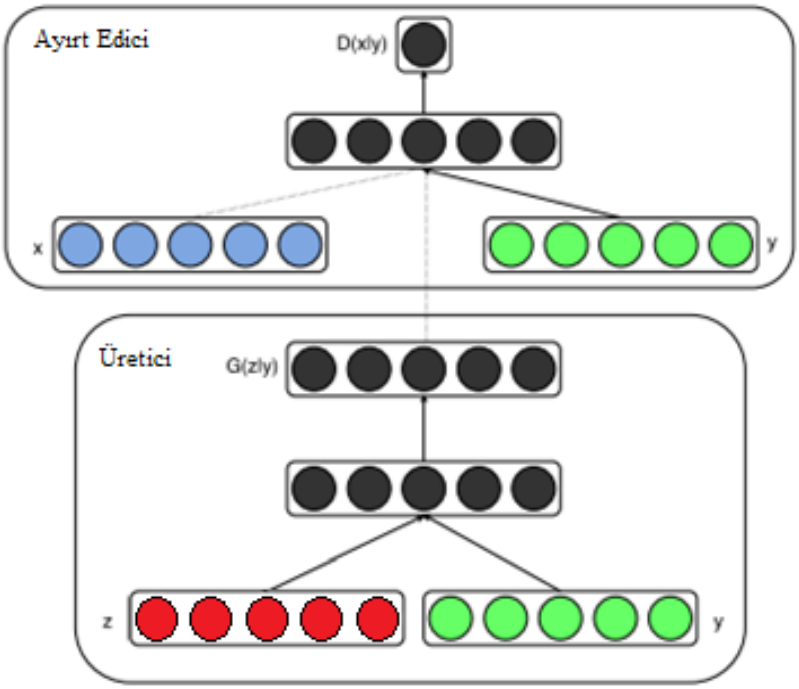


Şekil 13. DCGAN Üretici Ağ modeli (Radford vd., 2015)

Şekil 13'te de görüldüğü gibi Üretici (G) rasgele üretilen gürültü dizisi olan z verisini ilk olarak 4x4x1024 boyutunda yeniden şekillendirdikten sonra özellik evrişim katmanları ile sonuçta 64x64x3 boyutlu bir görüntüye dönüştürmektedir. Ayırt Edici ağ ise Üreticinin tersini, devrik evrişim kullanarak yapar ve karar katmanına çıkardığı özellikleri vererek 0 ila 1 arasında bir puanlama yapar. Bu puanlama sonucuna göre orijinal GAN modelindeki gibi hatalar hesaplanır, istenilen hata oranına gelene kadar ağdaki her katmanın ağırlığı güncellenerek eğitime devam edilir.

2.1.2. CGAN

CGAN, orijinal GAN modelindeki gibi rasgele sonuçlar üretmektense belli kriterleri sağlayan sonuçların üretilmesi üzerinedir. CGAN'ın ağ şeması Şekil 14'te ki gibidir (Mirza & Osindero, 2014).



Şekil 14. CGAN, Üretici ve Ayırt Edici ağ şeması (Mirza & Osindero, 2014)

Şekil 14'te görüldüğü gibi orijinal GAN modelinden farklı olarak girişe ayrıca *y* adında bir dizi eklenir. Bu dizini herhangi bir şekilde olabilir (Mirza & Osindero, 2014). Bir sınıflandırma etiketi eklenmesi durumunda ağ bu sınıflandırma kriterlerine bağlı bir öğrenme yapacak ve istenilen sonucun üretilmesini sağlayacaktır.

2.1.3. WGAN

Bu GAN modeli Denklem (1)'de kullanılan fonksiyon yerine daha stabil, daha az maliyetli bir öğrenme yöntemi olarak farklı bir fonksiyon önermiştir. Wasserstein Mesafesi olarak adlandırılan bu fonksiyon Denklem (15)'de görüldüğü gibidir (Arjovsky vd., 2017).

$$W(P_r, P_g) = \frac{inf}{\gamma \in \prod(P_r, P_g)} E_{(x,y)\sim\gamma}[\|x - y\|] \quad (15)$$

$\prod(\mathrm{P_r}, \mathrm{P_g})$ tüm eşzamanlı dağılımların kümesini temsil eder, burada marjinal dağılımlar sırasıyla Pr ve Pg'dir. Sezgisel olarak, $\gamma(\mathrm{x}, \mathrm{y})$, dağılımları Pr'yi Pg'ye dönüştürmek için x'ten y'ye taşınması gereken "kütle" miktarını gösterir. Wasserstein mesafesi, en iyi taşıma planının "maliyeti" dir (Arjovsky vd., 2017).

3. YAPILAN ÇALIŞMALAR

3.1. Ön İşlemler

Bu çalışma insan yüzlerinde belirlenen kurallara göre kadın/erkek gibi modifikasyonların yapılması üzerinedir. Bunun için CelebA (Liu vd., 2015) veri setinden bu türlere göre rasgele seçilen 10000 fotoğraf ile eğitim gerçekleştirilmiştir.

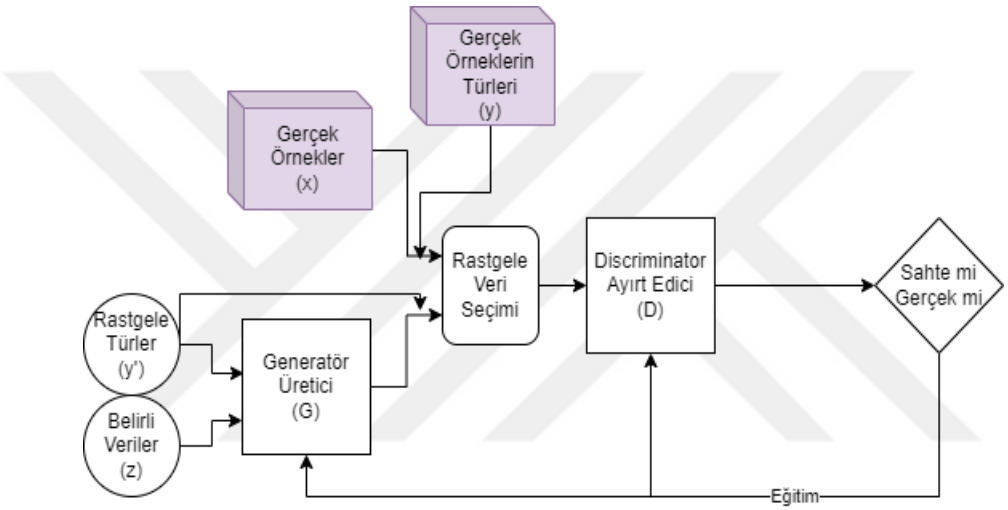
Bu veri setinin seçilmesindeki sebep çalışmada uygulayacağımız yöntemde fotoğraflar, etiket değerleri ile birleştirilerek eğitilmektedir. Bu yüzden çok çeşitli örneğin olduğu ve her bir örneğin belli etiketlerine sahip olduğu bu veri seti tercih edilmiştir. Bu veri setinin seçilmesindeki bir diğer sebebi ise 40 farklı türde etiketlenme yapılmış olmasıdır. Çok çeşitli etiket değerlerine sahip olması, farklı etiketler için ağın öğrenebilirliğini test etmemizi sağlamaktadır.

Aynı ağ farklı iki tür içinde eğitilmiştir. İlk tür temel amaç olan kadın/erkek olarak yüz görüntülerini değiştirmek içindir. Bunun için 5000 erkek, 5000 kadın fotoğrafı bu veri setinden rasgele seçilmiştir. Sakal ekleme türü için ise 5000 sakallı, 5000 sakalsız olmak üzere 10000 fotoğraf seçilmiştir. Eğitim için Python 3.9, Tensorflow 2.9, Cuda 11.6 ve Cudnn 8.1 sürümleri kurulmuştur. Tensorflow Keras Api kullanılarak kodlar yazılmıştır. Ön işlem olarak seçilen rasgele fotoğraflar 178x218 boyutunda olduğundan ortalanarak 178x178 şekilde kırpılmış ve 64x64'e sıkıştırılmıştır. Eğitim sırasında ayrıca rasgele düşük Gauss gürültüleri eklenilmiştir. Bu gürültüler az örnekler ile eğitim yaparken veri çeşitliğinin arttırılması ve Üretici-Ayırt Edici dengesinin sağlaması içindir.

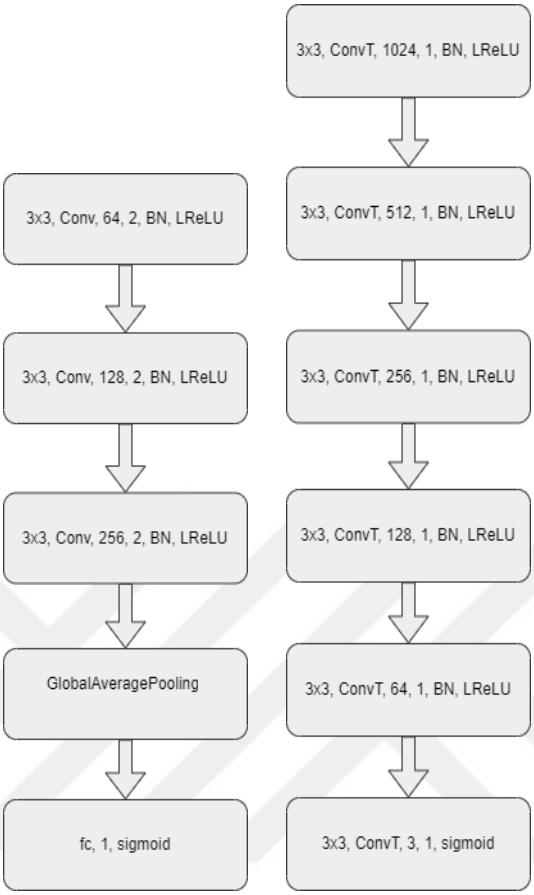
3.2. Uygulanan Yöntemler

Ağ yapısı olarak Ayırt Edici de evrişimli (convolutional) ağ ile üretilen görüntüler 64x64 ve rgb renk uzayında olup 3 katman evrişimli, 1 katman ortalama havuzlama ayrıca 1 katman tam bağlantılı (fullconnected) olarak tasarlanmıştır. Son katman hariç her katmanda aktivasyon fonksiyonu olarak LReLU kullanılmıştır. Son katmanda

Logistic/Sigmoid aktivasyon kullanıldı. Üretici ağ da ise giriş 64x64 rgb uzayında bir görüntü olup 6 katman devrik evrişim (convolution transpose) ve son katman hariç her katmanda aktivasyon fonksiyonu olarak LReLU kullanılmıştır. Üretici ağda son katmanda sigmoid aktivasyon kullanıldı. İki ağda da her katmanda filtre boyutu 3x3 olarak kullanılmıştır. Eğitimi daha stabil hale getirmek için evrişim ve devrik evrişim katmanlarından sonra normalizasyon uygulanmıştır. Yeni GAN modelinin eğitim sırasındaki yapısı Şekil 15'te görüldüğü gibidir. Şekil 16 (a) ve (b)'de uygun ağ şeması gözükmektedir.



Şekil 15. Oluşturulan yeni GAN modelinin çalışma şeması



(a) (b)

Şekil 16. Ağların şeması. (a) Ayırt Edici Ağ (b) Üretici Ağ

Orijinal GAN modelinden farklı olarak gürültü yerine modifiye edilecek fotoğraflar Üretici ağa, değiştirilmesi istenilen özellikler *y'* ile verilir. Üretici gelen örnekler ile koşul verilerini bir matris de birleştirerek istenilen türde fotoğrafı oluşturmaya çalışır. Bunun için Şekil 16 (b)'de ki Üretici ağ yapısı kullanıldı. Üretici tarafından üretilen görüntüler ile istenilen türler çıkışta birleştirilir, aynı durum gerçek örnekler üzerinde de uygulanır. Sonra bu veriler karıştırılarak ağ yapısı Şekil 16 (a)'da ki gibi olan Ayırt Edici ağa gönderilir. Bu ağ gelen örneklerin gerçek olmasına ve ayrıca verilen türle uyumlu olup olmadığına 0 ila 1 arası bir puanlama ile karar verir. "0" durumu uyumsuz veya sahte anlamına gelirken, "1" durumu uyumlu ve gerçek anlamına gelmektedir. Ayırt Edici için hata, gerçek görüntülere ne kadar sahte dediği ve sahte görüntüleri ne kadar gerçek olarak tahmin ettiğinin toplamıdır.

Üretici için ise hata, ürettiği görüntülerin Ayırt Edici tarafından ne kadar sahte olarak fark edildiğinin oranı ve görüntünün tekrar tersini aldığında orijinal görüntü ile aynı olmama oranı kadardır. Bu benzerlik oranı MAE hata fonksiyonu ile hesaplanmaktadır. Bu işlemlerden sonra eğitim aşamasına geçilerek Geri Yayılım Algoritmaları ile iki ağ, bu hata oranlarına göre yeniden ağırlıklandırılır. Bu işlem her döngüde tekrar edilir. Ayırt Edici ağ artık gerçek görüntüler ile sahte görüntüleri ayırt edemediğinde eğitim tamamlanmış olur. Bu eğitim Algoritma 2'de ki gibidir.

Algoritma 2:

Ortalama, kırpma ve normalize etme sonucu N adet görüntü $\{x^{(1)}, \dots, x^{(N)}\}$

Koşul verilerini normalize etme sonucu N adet koşul $\{c^{(1)}, \dots, c^{(N)}\}$

Gerçek veriler ile birleştirilen N adet görüntü-koşul verisi $\{xc^{(1)}, \dots, xc^{(N)}\}$

Eğitim için N adet görüntü-terslenmiş koşul verisi $\{x(1-c)^{(1)}, \dots, x(1-c)^{(N)}\}$

for "eğitim sayısı" do:

for "görüntü sayısının parti boyutuna bölümü" do:

Üretici ile m adet $\{x(1-c)^{(1)}, \dots, x(1-c)^{(m)}\}$ görüntü oluştur:

$\{z^{(1)}, \dots, z^{(m)}\}$

Ayırt Edici de hata hesapla:

Ayırt Edici Hata = BCE ($\{z(1-c)^{(1)}, \dots, z(1-c)^{(m)}\}$) +

BCE ($\{xc^{(1)}, \dots, xc^{(m)}\}$)

Üreticide görüntüleri $\{z^{(1)}, \dots, z^{(m)}\}$ kullanarak eski haline getir:

$\{x'^{(1)}, \dots, x'^{(m)}\}$

Üretici de hata hesapla:

Üretici Hata = BCE ($1 - \{z(1-c)^{(1)}, \dots, z(1-c)^{(m)}\}$) +

MAE ($\{|x'-x|^{(1)}, \dots, |x'-x|^{(m)}\}$)

Ayırt Edici hatası ile Adam optimizasyon yöntemi algoritması kullanarak ağırlıkları güncelle:

Adam (öğrenme_oranı=0.0002, Beta1=0.5, Ayırt Edici Hata)

Üretici hatası ile Adam optimizasyon yöntemi algoritması kullanarak ağırlıkları güncelle:

Adam (öğrenme_oranı=0.0002, Beta1=0.5, Üretici Hata)

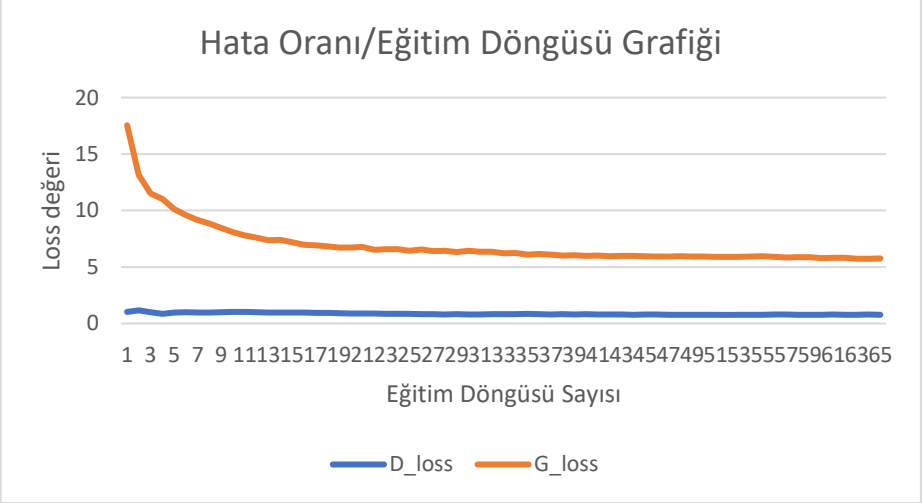
end for

end for

m parti(batch) boyutunu göstermektedir. N eğitimde kullanılan tüm görüntülerin sayısı

4. İRDELEME

Çoğu GAN modelinden farklı olarak rasgele gürültüler ile rasgele sonuçlar üretmektense gerçek fotoğraflar üzerinde istenilen durumların oluşturulması bu çalışmanın temel amacıdır. Bu modele benzer CNN ağları bulunmakla birlikte çoğu ticari amaçlı olduğundan model yapısı paylaşılmamıştır. Ayrıca bu çalışmada hiçbir morfolojik filtre de kullanılmadan sakallı/sakalsız ile kadın/erkek durumlarını öğrendiği görülmektedir. Bu süreç, sadece fotoğraflar arasında ortak benzerliklerin ağ tarafından fark edilmesi ile gerçekleştirilmiştir. Bu sayede sadece fotoğraf ve o fotoğrafın sakallı/sakalsız, erkek/kadın bilgisinin etiketi eğitim için yeterli olmaktadır.



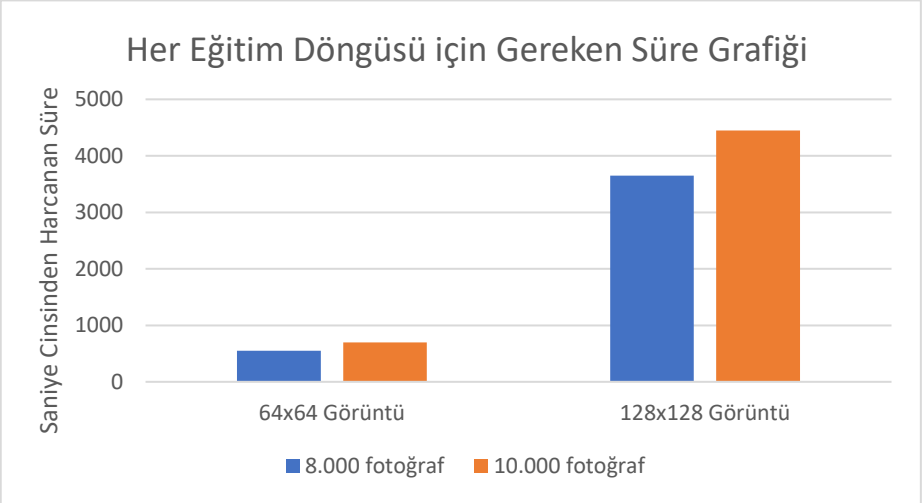
Şekil 17. Eğitim döngüsü sayısına göre hata oranlarının grafiği. G ve D kayıp oranları ikili çapraz entropi ile hesaplanmıştır.

Şekil 17'de ilk 65 eğitim döngüsü için kayıp değerlerinin grafiği gözükmektedir. Ayırt Edici hata oranı D_loss, Üreticinin hata oranı G_loss olarak gösterilmektedir. Grafikten de anlaşıldığı üzere eğitim sayısı arttıkça Üretici hata oranı ilk başta hızlıca azalırken ilerleyen

eğitim döngülerinde giderek daha az oranda azalmaktadır. Bunun sebebi Üreticinin 2 hata hesap fonksiyonunun toplamı olduğundan ve 2. Fonksiyon olan Ortalama Mutlak Hata fonksiyonu her döngüde daha hızlı bir şekilde 0'a yaklaştığından dolayı olduğu gözlemlenmiştir. İlk hata hesabı fonksiyonu olan ikili çapraz entropi fonksiyonunun sonucu Üretici ve Ayırt Edicide yavaş yavaş azalmaktır.

5. DENEYSEL SONUÇLAR

Truba uzak bilgisayarlarında Şekil 16 (a) ve (b) de görünen modeller kullanılarak oluşturulan ağlar yaklaşık olarak her eğitim döngüsünde 64x64 için 700 saniye harcamıştır. Aynı boyuttaki 8000 fotoğraf ile eğitimde ise 550 saniye ortalama süre harcanmıştır 2 durumda da batch büyüklüğü 32 seçilmiştir. 128x128'lik eğitim için batch (parti) büyüklüğü 4 seçilmiş olup 8000 fotoğraf ile ortalama 3650 saniye her bir eğitim döngüsü olup 10000 fotoğrafla ise 4450 saniye sürmüştür. Şekil 18'de bu sürelerin karşılaştırıldığı grafik görünmektedir. Parti büyüklüğünün görselin boyutuna göre değişmesinin nedeni bilgisayarda yeterli vram bulunamamasıdır. 64x64'lük 32 batch size yaklaşık 13gb vram kullanmaktadır. 128x128'lik 4 batch size eğitimde ise yaklaşık 14gb vram kullanılmaktadır.

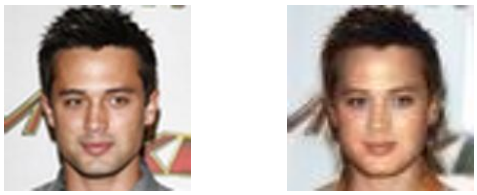


Şekil 18. Eğitimde kullanılan görüntü boyutu ve sayısına göre eğitim süreleri

Sakal ekleme için eğitilen ağın yaklaşık 104 eğitim döngüsü için verdiği bazı örnek sonuçlar Şekil 19'da gözükmektedir. Şekil 20'de aynı ağın kadın/erkek değişimi için tekrar eğitimi sonucu oluşan sonuçlar görülmektedir.

(a) (b)

Şekil 19. Deneysel sonuçlar. (a) Girişten verilen fotoğraf (b) Sakallı sonuç



(a) (b)

Şekil 20. Erkek/Kadın değiştirme sonuçları. (a) Verilen fotoğraf (b) Dönüştürülen Sonuç

(a) (b) (c)

Şekil 21. Geri eski haline getirme sonucu. (a) Verilen fotoğraf (b) Dönüştürülen sonucu (c) Tekrar eski haline getirme

Şekil 21'de Denklem (15)'de ki Yapısal benzerlik indeksi ölçüsü diğer bir adıyla SSIM (Otero vd., 2021) fonksiyonu kullanılarak %90 oranında fotoğrafı eski haline getirilmiştir.

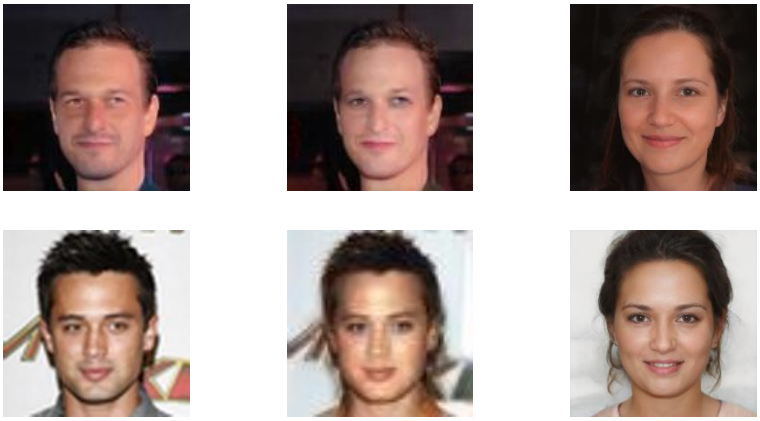
$$SSIM(x,y) = \left(\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}\right)\left(\frac{2\sigma_{xy}+C_2}{\sigma_x^2+\sigma_y^2+C_2}\right) \quad (15)$$

Şekil 22'de başka bir fotoğraf üzerinde Erkek/Kadın değiştirme ve geri alma sonucu gösterilmektedir.

(a) (b) (c)

Şekil 22. Bir diğer Erkek/Kadın değiştirme ve eski haline getirme sonucu.
(a) Verilen fotoğraf (b) Dönüştürülen sonucu (c) Tekrar eski haline getirme

Şekil 23'te ve 24'te AILab (URL-4, 2023) tarafından geliştirilen ağın sonuçları ile oluşturduğumuz yeni ağın sonuçlarının karşılaştırılması bulunmaktadır. Bu tarz ticari uygulamalarda ağ yapısı gizli olmaktadır. Bu yüzden doğrudan ağ yapılarına göre bir karşılaştırma yapmak mümkün değildir.



(a) (b) (c)

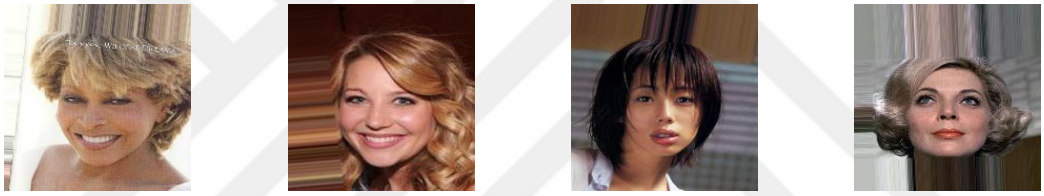
Şekil 23. Oluşturulan ağ ile ticari bir uygulamanın karşılaştırılmasının sonucu
(a) Verilen fotoğraf (b) Oluşturulan sonuç (c) AILab sonucu (URL-4, 2023)

AILab gibi ağları gizli ve ticari kullanımı olan uygulamalar kendi özel veri setleri ile eğitim yapmakta ve daah gelişmiş bilgisayarlarda daha geniş kapsamlı ağlar oluşturarak sonuçlarını daha gerçekçi yapması mümkündür. Fakat ağ yapısı herkese açık olmadığı için çalışmada oluşturduğumuz ağ ile arasındaki farklar gösterilememektedir.

6. ÖNERİLER

Ağ geliştirilerek 128x128 ya da 256x256'lık görüntüler üzerinde de çalışılabilir. Bunun için daha yüksek kapasiteli bilgisayarlar ve daha fazla örnek üzerinde eğitim yapılabilir.

Farklı veri seti kullanılabilir. Bu veri setinde az sakalı olan ile çok sakallı örneklerin ikisi de aynı ağırlık değerine sahip olduğu görülmüştür. Ayrıca Bazı fotoğrafların bozuk olduğu da tespit edilmiştir. Bu da eğitimi zorlaştırmaktadır. Şekil 24'te bazı bozuk fotoğraflar gösterilmektedir.



Şekil 24. Veri setinde bulunan bazı hatalı fotoğraflar

Ayrıca bu veri setinde maksimum görüntü boyutu 178x218 olduğundan 256x256'lık eğitim için de farklı veri seti ihtiyacı bulunmaktadır.

Üretici ve Ayırt Edici ağ arasında denge durumunu sağlamak için farklı eğitim yöntemleri kullanılabilir. Örneğin WGAN (Arjovsky vd., 2017) benzeri farklı yöntemler kullanılabilir.

7. KAYNAKLAR

Arjovsky, M., Chintala, S., & Bottou, L. (2017, July). Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214-223). PMLR.

Boué, L. (2018). Deep learning for pedestrians: backpropagation in CNNs. arXiv preprint arXiv:1811.11987.

Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., & Dahl, G. E. (2019). On empirical comparisons of optimizers for deep learning. arXiv preprint arXiv:1910.05446.

Dorfer, M., Kelz, R., & Widmer, G. (2015). Deep linear discriminant analysis. arXiv preprint arXiv:1511.04707.

Gershenson, C. (2003). Artificial neural networks for beginners. arXiv preprint cs/0308031.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.

Hinton, G., Srivastava, N., & Swersky, K., (2020) Lecture 6a: Overview of Mini-Batch Gradient Descent. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Karabayır İ. (2018). Gradyan ve Özel Bir Hiper Düzlem Temelli Yeni Bir Optimizasyon Algoritması: Evriştirilmiş Gradyan Yönü ile Optimizasyon. İstanbul Üniversitesi. Sosyal Bilimler Enstitüsü. Doktora Tezi. İstanbul http://nek.istanbul.edu.tr:4444/kos/TEZ/57788.pdf

Kılıçarslan, S. , Adem, K. & Çelik, M. (2021). An overview of the activation functions used in deep learning algorithms . Journal of New Results in Science , 10 (3) , 75-88 . DOI: 10.54187/jnrs.1011739

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Lederer, J. (2021). Activation functions in artificial neural networks: A systematic overview. arXiv preprint arXiv:2101.09957.

Liu, Ziwei and Luo, Ping and Wang, Xiaogang and Tang, Xiaoou (2015). Proceedings of International Conference on Computer Vision (ICCV). Deep Learning Face Attributes in the Wild

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., & Paul Smolley, S. (2017). Least squares generative adversarial networks. In Proceedings of the IEEE international conference on computer vision (pp. 2794-2802).

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Otero, D., La Torre, D., Michailovich, O., & Vrscay, E. R. (2021). Optimization of structural similarity in mathematical imaging. Optimization and Engineering, 22, 2367-2401.

Park, H., Yoo, Y., & Kwak, N. (2018). Mc-gan: Multi-conditional generative adversarial network for image synthesis. arXiv preprint arXiv:1805.01123.

Qi, J., Du, J., Siniscalchi, S. M., Ma, X., & Lee, C. H. (2020). On mean absolute error for deep neural network based vector-to-vector regression. IEEE Signal Processing Letters, 27, 1485-1489.

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

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Salehi, P., Chalechale, A., & Taghizadeh, M. (2020). Generative adversarial networks (GANs): An overview of theoretical model, evaluation metrics, and recent developments. arXiv preprint arXiv:2005.13178.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. Towards Data Sci, 6(12), 310-316.

Szandała, T. (2021). Review and comparison of commonly used activation functions for deep neural networks. Bio-inspired neurocomputing, 203-224.

Turhan, M. M. & Talu, M. F. (2022). Adaptif Sigmoid, Lojistik Sigmoid ve Tanjant Hiperbolik Aktivasyon Fonksiyonların Tam Bağlı ve Konvolüsyonel Sinir Ağlarında Kıyaslanması. Muş Alparslan Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 3 (2),89-101. Retrieved from https://dergipark.org.tr/tr/pub/maummfd/issue/74679/1223551

URL-1. (2023, Mart 10) Module:tf.keras.losses: https://www.tensorflow.org/api_docs/python/tf/keras/losses adresinden alınmıştır.

URL-2. (2023, Mart 10) Module:tf.keras.losses: https://keras.io/api/losses/ adresinden alınmıştır.

URL-3. (2023, Mayıs 20) Convolutional Neural Networks for Visual Recognition: https://cs231n.github.io/neural-networks-3/ adresinden alınmıştır.

URL-4. (2023, Temmuz 4) Swap Gender to See the Other You: https://ailab.wondershare.com/tools/gender-swap-filter.html adresinden alınmıştır.

Ven, L., & Lederer, J. (2021). Regularization and Reparameterization Avoid Vanishing Gradients in Sigmoid-Type Networks. arXiv preprint arXiv:2106.02260.

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.

ÖZGEÇMİŞ

2014 yılında Beşikdüzü Anadolu Lisesinden mezun oldu. 2020 yılında Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü Lisans derecesini tamamladı. 2020'den günümüze kadar Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Yüksek Lisans bölümünde okumaktadır.

Araştırma alanı Yapay Zekâ ve Üretken Çekişmeli Ağlar'ı içermektedir.

1. Kardal, E., & Nabiyev, V. (2023). GAN ile İnsan Yüzlerinde Değişiklik Yapma. *International Conference on Pioneer and Innovative Studies*, *1*, 498–500. https://doi.org/10.59287/icpis.879