

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇEKİŞMELİ ÜRETİCİ AĞLAR İLE SIFIRDAN İKİ
BOYUTLU KARAKTER ÜRETİMİ VE
KARAKTER RENKLENDİRİLMESİNDE
PIXELSIGHT BLOKLARININ KULLANIMIYLA
ÇEŞİTLİLİK VE KALİTENİN ARTIRILMASI

Volkan Can BACAKSIZ

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

Danışman
Dr. Öğr. Üyesi Oğuz ALTUN

Mart, 2025

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

**ÇEKİŞMELİ ÜRETİCİ AĞLAR İLE SIFIRDAN İKİ
BOYUTLU KARAKTER ÜRETİMİ VE KARAKTER
RENKLENDİRİLMESİNDE PIXELSIGHT BLOKLARININ
KULLANIMIYLA ÇEŞİTLİLİK VE KALİTENİN
ARTIRILMASI**

Volkan Can BACAŞIZ tarafından hazırlanan tez çalışması 18.03.2025 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programı **YÜK-SEK LİSANS TEZİ** olarak kabul edilmiştir.

Dr. Öğr. Üyesi Oğuz ALTUN
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Dr. Öğr. Üyesi Oğuz ALTUN, Danışman
Yıldız Teknik Üniversitesi

Prof. Dr. Gökhan BİLGİN, Üye
Yıldız Teknik Üniversitesi

Prof. Dr. Selçuk SEVGİN, Üye
İstanbul Üniversitesi - Cerrahpaşa

Danışmanım Dr. Öğr. Üyesi Oğuz ALTUN sorumluluğunda tarafımca hazırlanan Çekişmeli Üretici Ağlar ile Sıfırdan İki Boyutlu Karakter Üretimi ve Karakter Renklendirilmesinde PixelSight Bloklarının Kullanımıyla Çeşitlilik ve Kalitenin Artırılması başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Volkan Can BACAKSIZ

İmza

Aileme...



TEŞEKKÜR

Tez çalışmamın hazırlanmasında bilgi, destek ve rehberliği için Sayın Dr. Öğr. Üyesi Oğuz ALTUN'a sonsuz teşekkürlerimi sunarım. Tez jürimde yer alan Sayın Prof. Dr. Gökhan BİLGİN ve Sayın Prof. Dr. Selçuk SEVGİN hocalarıma değerli görüşleri ve önerileri için şükranlarımı sunarım. Tez araştırmam boyunca veri sağlamamda destek olan Yıldız Teknik Üniversitesi'ne de ayrıca teşekkür ederim. Son olarak tüm bu süreçte maddi ve manevi olarak bana her türlü desteği sağlayan aileme teşekkür ederim.

Volkan Can BACAĞIZ

İÇİNDEKİLER

SİMGE LİSTESİ	viii
KISALTMA LİSTESİ	ix
ŞEKİL LİSTESİ	xi
TABLO LİSTESİ	xv
ÖZET	xvii
ABSTRACT	xix
1 GİRİŞ	1
1.1 2 Boyutlu (2B) Çizim	1
1.1.1 Piksel Sanatı	1
1.2 Karakter Üretim Süreci	3
1.2.1 Karakter Üretimi	3
1.2.2 Karakter Renklendirmesi	3
1.2.3 Karaktere Ait Animasyon Oluşturma	4
1.3 Karakter Üretiminin Otomasyonlaştırılması	5
2 YÖNTEMLER	7
2.1 Optimizasyon Yöntemleri	8
2.2 Çekişmeli Üretici Ağ (ÇÜA) Mimarileri	9
2.2.1 Klasik Çekişmeli Üretici Ağ (ÇÜA)	9
2.2.2 Derin Konvolüsyonel Çekişmeli Üretici Ağ (DKÇÜA)	11
2.2.3 Koşullu Çekişmeli Üretici Ağ (KÇÜA)	13
2.2.4 Pix2Pix	14
2.2.5 Seg2Pix	17
2.2.6 Çok Ayırıcılı Çekişmeli Üretici Ağ (ÇAÇÜA)	18
2.2.7 Yaratıcı Çekişmeli Üretici Ağ (YÇÜA)	20
2.2.8 Stil Çekişmeli Üretici Ağ 2 (StyleGAN2)	21
2.2.9 Bağlamsal Çekişmeli Üretici Ağ (BÇÜA)	23
2.2.10 Döngü Tutarlı Çekişmeli Üretici Ağ (CycleGAN)	25

2.2.11	Alanlar Arası İlişkileri Keşfeden Çekişmeli Üretici Ağ (DiscoGAN)	27
2.2.12	Yıldız Çekişmeli Üretici Ağ (StarGAN)	30
2.3	Varyasyonel Otomatik Kodlayıcı (VOK) Mimarileri	32
2.3.1	Klasik Varyasyonel Otomatik Kodlayıcı (VOK)	32
2.3.2	Vektör Kuantize Edilmiş Varyasyonel Otomatik Kodlayıcı (VK-VOK)	33
2.3.3	Piksel Vektör Kuantize Edilmiş Varyasyonel Otomatik Kodlayıcı (Pixel VQ-VAE)	34
2.3.4	Koşullu Varyasyonel Otomatik Kodlayıcı (KVOK)	35
3	DENEYLER	37
3.1	Karakter Renklendirme	37
3.1.1	Kullanılacak Veri Setinin Hazırlanması	38
3.1.2	Mimarilerin Karşılaştırılması	40
3.1.3	Sorunların Tanımlanması	42
3.1.4	İyileştirmeler	44
3.1.5	Gerçek Dünya Verileriyle Test Süreci	47
3.2	Karakter Üretimi	51
3.2.1	Deney Süreci ve İyileştirmeler	51
3.2.2	Kullandığımız Parametreler	52
3.2.3	Kullanıcı Aracılığı ile Değerlendirme	53
4	SONUÇ	55
	KAYNAKÇA	57
A	Karakter Renklendirme Deneyinin Detayları	64
A.1	Değerlendirme	64
A.2	Mimari	65
A.2.1	Üretici Ağ	65
A.2.2	Ayırıcı Ağ	66
A.3	Mimarilere Ait Kodlama Detayları	66
B	Karakter Üretimi Deneyinin Detayları	69
B.1	Mimari Detayları	69
B.1.1	Mimarilerin Görselleştirilmesi	69
B.1.2	Mimarilere Ait Kodlamalar	70
B.2	Ankete İlişkin Detaylar	71
B.2.1	1. Form	71
B.2.2	2. Form	73

B.2.3	3. Form	75
B.2.4	4. Form	77
B.2.5	5. Form	79
C	Terminoloji	82
	TEZDEN ÜRETİLMİŞ YAYINLAR	89



SİMGE LİSTESİ

D	Ayırıcı ağ
$\mathbb{E}$	Beklenen değer
m	Birinci momentum
g	Gradyan
v	İkinci momentum
$\mathcal{L}$	Kayıp fonksiyonu, maliyet fonksiyonu
$\mathbf{M}$	Maske matrisi
μ	Ortalama
σ	Standart sapma
G	Üretici Ağ

KISALTMA LİSTESİ

2B	2 Boyutlu
3B	3 Boyutlu
AdaIN	Uyarlanabilir Örnek Normalizasyon
AYV	Anime Yüz Veri Seti
BÇÜA	Bağlamsal Çekişmeli Üretici Ağ
BS	Başlangıç Skoru
ÇAÇÜA	Çok Ayırıcılı Çekişmeli Üretici Ağ
ÇÜA	Çekişmeli Üretici Ağ
CycleGAN	Döngü Tutarlı Çekişmeli Üretici Ağ
DiscoGAN	Alanlar Arası İlişkileri Keşfeden Çekişmeli Üretici Ağ
DKÇÜA	Derin Konvolüsyonel Çekişmeli Üretici Ağ
ELU	Üssel Doğrusal Birim
FBM	Fréchet Başlangıç Mesafesi
KÇÜA	Koşullu Çekişmeli Üretici Ağ
KL	Kullback-Leibler
Leaky ReLU	Sızıntılı Düzeltilmiş Doğrusal Birim
MÖPİÜ	Makine Öğrenimi ile Prosedürel İçerik Üretimi
OKH	Ortalama Karesel Hata
OMH	Ortalama Mutlak Hata
ReLU	Düzeltilmiş Doğrusal Birim
ResNet	Artık Sinir Ağı
PİÜ	Prosedürel İçerik Üretimi
Pixel VQ-VAE	Piksel Vektör Kuantize Edilmiş Varyasyonel Otomatik Kodlayıcı

SGİ	Stokastik Gradyan İniş
StarGAN	Yıldız Çekişmeli Üretici Ağ
StyleGAN	Stil Çekişmeli Üretici Ağ
StyleGAN2	Stil Çekişmeli Üretici Ağ 2
VK-VOK	Vektör Kuantize Edilmiş Varyasyonel Otomatik Kodlayıcı
VOK	Varyasyonel Otomatik Kodlayıcı
YÇÜA	Yaratıcı Çekişmeli Üretici Ağ



ŞEKİL LİSTESİ

Şekil 1.1	(a) 2B karakter yüzü. (b) Piksel sanatı karakter.	2
Şekil 1.2	Oyun karakteri hareketli çizimi. Şekil 1.1.b’de gösterilen karaktere ait hareketli çizimdir.	4
Şekil 2.1	Çekişmeli üretici ağ mimarisine genel bakış.	10
Şekil 2.2	Derin konvolüsyonel çekişmeli üretici ağ mimarisine genel bakış.	12
Şekil 2.3	Koşullu çekişmeli üretici ağ mimarisine genel bakış.	13
Şekil 2.4	U-Net üretici ağı.	15
Şekil 2.5	Yama tabanlı ayırıcı ağ. Görüntüdeki küçük parçalardan gerçek ve sahte tahminleri (predictions) yapılır.	15
Şekil 2.6	Seg2Pix mimarisinin kullanımına genel bakış. Seg2Pix mimarisi 2 farklı işlem için kullanılır. İlk işleminde segment görüntü elde edilir. İkinci işlemde ise istenilen çıktı olan boyanmış görüntü elde edilir.	17
Şekil 2.7	Çok ayırcılı çekişmeli üretici ağ mimarisine genel bakış.	19
Şekil 2.8	Yaratıcı çekişmeli üretici ağ mimarisine genel bakış. Yaratıcı çekişmeli üretici ağ mimarisinde özellikle ayırıcı ağın işlevi çok önemlidir. Ayırıcı ağ, kendisine verilen üretici ağ çıktısının sanat olup olmadığını değerlendirir. Aynı zamanda üretilen görüntünün hangi sanat stiline (art style) ait olduğunu sınıflandırır.	20
Şekil 2.9	(a) StyleGAN mimarisindeki üretici ağ. Üretici ağ içerisinde normalizasyon işlemi modülasyondan hemen sonra gerçekleşir. (b) StyleGAN2 mimarisindeki üretici ağ. Yapılan iyileştirme ile beraber normalizasyon işlemi ile modülasyon işleminin bağlantısı koparılır.	22
Şekil 2.10	Bağlamsal çekişmeli üretici ağ mimarisine genel bakış. Bağlamsal çekişmeli üretici ağ mimarisinde girdi olarak kullanılan görüntüde el çizimi ile üretilecek kısım birleştirilerek verilir.	24
Şekil 2.11	CycleGAN mimarisine genel bakış.	26
Şekil 2.12	DiscoGAN mimarisine genel bakış.	28
Şekil 2.13	StarGAN mimarisinde kullanılan üretici ağ. Üretici ağ içerisinde farklı alanlar arasında bağlantı kurulur.	30

Şekil 2.14	Varyasyonel otomatik kodlayıcı mimarisine genel bakış.	33
Şekil 2.15	Vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisine genel bakış.	34
Şekil 2.16	Piksel vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisine genel bakış.	35
Şekil 2.17	Koşullu varyasyonel otomatik kodlayıcı mimarisine genel bakış. .	36
Şekil 3.1	Çalışmamızın akış diyagramı.	37
Şekil 3.2	Yeniden ölçeklendirilen ve üretilen çizgi sanatı görüntüleri. (a) Veri setinin orta sıralarında bulunan örnek görüntü. (b) Veri setinin sonlarına doğru örnek görüntü.	39
Şekil 3.3	Kıyaslama yaparken yaptığımız işlemlerin kısa bir özetini içeren şema. Mimariğin girdilerinin daha iyi anlaşılmasını sağlayacaktır. Jiang ve Sweetser tarafından ortaya atılan Pix2Pix mimarisini kullanırken yaptıkları çalışmadaki gibi, YUV renk uzayının Y kanalından çizgi sanatı üretme yönteminden yararlanmadık. Bunun yerine ürettiğimiz çizgi sanatını YUV renk uzayına çevirerek kullandık. Bunu deneylerimizde girdi olarak büyük farklılıkların önüne geçmek için yaptık.	39
Şekil 3.4	Mimarilerin orijinal görüntüye benzerlik anlamında ürettiği en iyi çıktılar. OKH değerlendirme yöntemi baz alınmıştır (Tablo 3.1). .	40
Şekil 3.5	Isola vd. tarafından ortaya konan Pix2Pix mimarisinin orijinal görüntüye yakın bir görüntü ürettikten sonra, ilerleyen adımlarda ürettiği görüntüler.	41
Şekil 3.6	(a) Eğitim verisinin HSV renk uzayında ton parametresine ait histogramı. (b) HSV renk uzayında doygunluk ve değer parametrelerini göz ardı ettiğimizde ton parametresinin bir döngü içerisinde olduğu bilinmektedir. Ton (H) parametresi 360 değeri ile beraber aslında başa dönmektedir. Histogramın daha kolay okunabilmesi amacıyla okunabilirliğe göre düzenleme yaptık ve çok az pikselde bulunan ton parametresinin değerlerini histogramdan kaldırdık. (c) Eğitimin başlangıcında üretilen görüntüler. Renk baskınlığının eğitimin ilk adımlarındaki etkisini ortaya koymak için paylaştık.	42

Şekil 3.7	(a) Orijinal Pix2Pix Mimarisi. (b) Ortaya koyduğumuz Pix2Pix modifikasyonu (Line2Pix). Bizim eklediğimiz katmanları kırmızı renkle renklendirdik. Genel olarak kodlamayı Pix2Pix mimarisinden aldık. Mimarimizi daha iyi anlaşılabilmesi için görsel olarak ortaya koyduk. Yaptığımız eklemelerde çekirdek boyutu (kernel size) olarak 1x1, adım (stride) olarak 1 değerini kullandık.	44
Şekil 3.8	Mimarilerin eğitimin başlangıç aşamalarındaki renk değişimleri. Ortaya koyduğumuz Line2Pix mimarisinin renklendirmeye başlamak için ihtiyaç duyduğu eğitim süresi daha fazladır. Minimal değişikliklere izin vermemiz sonucu eğitim süreleri uzamıştır.	45
Şekil 3.9	Yaptığımız iyileştirmeler sonucunda orijinal görüntüye yaklaşılan üretim adımından sonraki adımlarda üretilen örnek görüntüler. . .	46
Şekil 3.10	Yaptığımız iyileştirmeler sonucunda 2 farklı görüntümüz için OKH yöntemine göre üretilmiş en iyi görüntüler. Görüntülere ait OKH skorları Tablo 3.2’de paylaşılmıştır. (a) Orijinal Pix2Pix mimarisi ile üretilen görüntüde yüz bölgesinde mavi renk tonları bulunmaktadır. (b) Line2Pix mimarimiz ile üretilen görüntünün hem yüz rengi, hem de saç rengi orijinal görüntüye daha yakındır.	46
Şekil 3.11	Bize ait olan el çizimleri.	48
Şekil 3.12	Bize ait olan el çizimlerini (Şekil 3.11) test verisi olarak kullandığımızda ürettiğimiz sonuçlar. Girdi olarak kullandığımız kağıt ve bilgisayar çizimleri Şekil 3.11’de görülebilir. Kağıt girdisi ile üretilen sonuçlara ait OKH skorlarını Tablo 3.3’te, bilgisayar girdisine ait OKH skorlarını Tablo 3.4’te paylaştık. Mimarilerin eğitiminde uyarlanabilir eşik yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.	48
Şekil 3.13	Yeni çizgi sanatı görüntüsü oluşturma yöntemimiz ve kenar algılama yöntemleri.	50
Şekil 3.14	Mimarileri yeni çizgi sanatı oluşturma yöntemimizle elde ettiğimiz çizgi sanatı görüntülerle eğittikten sonra, kağıt ve bilgisayar çizimlerini girdi olarak verdiğimizde mimarilerden elde ettiğimiz görüntüler. Girdi olarak kullandığımız kağıt ve bilgisayar çizimleri Şekil 3.11’de görülebilir.	50
Şekil 3.15	(a) DKÇÜA mimarisine genel bakış. (b) Karakter üretimi için ortaya koyduğumuz mimarimize genel bakış.	52
Şekil 3.16	Karakter üretimi için kullandığımız mimarilerden elde ettiğimiz örnek çıktılar.	53

Şekil A.1	Isola vd. Pix2Pix mimarisinde kullanılan üretici ağ.	65
Şekil A.2	Ortaya koyduğumuz Line2Pix mimarisinde kullanılan üretici ağ. Eklemelerimiz kırmızı renkle renklendirilmiştir.	66
Şekil A.3	Isola vd. Pix2Pix mimarisinde kullanılan ayırıcı ağ.	67
Şekil A.4	Ortaya koyduğumuz Line2Pix mimarisinde kullanılan ayırıcı ağ. Eklemelerimiz kırmızı renkle renklendirilmiştir.	67
Şekil B.1	Kullandığımız DKÇÜA mimarisi.	69
Şekil B.2	Ortaya koyduğumuz DKÇÜA mimarisi. Eklemelerimiz kırmızı renkle renklendirilmiştir.	70
Şekil B.3	1. formda kullanılan karakter grupları.	72
Şekil B.4	2. formda kullanılan karakter grupları.	74
Şekil B.5	3. formda kullanılan karakter grupları.	76
Şekil B.6	4. formda kullanılan karakter grupları.	78
Şekil B.7	5. formda kullanılan karakter grupları.	80

TABLO LİSTESİ

Tablo 3.1	Mimarilere ait OKH skorları. MANGAN mimarisinin ürettiği görüntüler çok gürültülü olduğu için dahil edilmemiştir.	40
Tablo 3.2	Şekil 3.10’da paylaştığımız görüntülere ait OKH skorları. Aynı zamanda 4 farklı test görüntümüz için aldığımız OKH skorlarının ortalamasını paylaştık.	46
Tablo 3.3	Mimarilere kağıt girdisi (Şekil 3.11) verilerek üretilen görüntülere ait OKH skorları. Mimarilerin eğitiminde uyarlanabilir eşik yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.	48
Tablo 3.4	Mimarilere bilgisayar girdisi (Şekil 3.11) verilerek üretilen görüntülere ait OKH skorları. Mimarilerin eğitiminde uyarlanabilir eşik yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.	49
Tablo 3.5	Mimarilere kağıt girdisi (Şekil 3.11) verilerek üretilen görüntülere ait OKH skorları. Mimarilerin eğitiminde yeni çizgi sanatı oluşturma yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.	50
Tablo 3.6	Mimarilere bilgisayar girdisi (Şekil 3.11) verilerek üretilen görüntülere ait OKH skorları. Mimarilerin eğitiminde yeni çizgi sanatı oluşturma yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.	51
Tablo 3.7	Yaptığımız ankete ait sonuçlar.	54
Tablo A.1	Bölüm 3.1.2’de kullandığımız mimarilerin ürettiği çıktıları ait FBM skorları. 20.000 - 40.000 adım aralığındaki çıktıları kullandık.	64
Tablo A.2	Mimarimize ait FBM skoru. 20.000 - 40.000 adım aralığındaki çıktıları kullanıyoruz.	65
Tablo B.1	Şekil B.3’te bulunan karakterlerin ait olduğu mimariler.	73
Tablo B.2	1. forma ait sonuçlar.	73
Tablo B.3	Şekil B.4’te bulunan karakterlerin ait olduğu mimariler.	75
Tablo B.4	2. forma ait sonuçlar.	75
Tablo B.5	Şekil B.5’te bulunan karakterlerin ait olduğu mimariler.	77

Tablo B.6	3. forma ait sonuçlar.	77
Tablo B.7	Şekil B.6’da bulunan karakterlerin ait olduğu mimariler.	79
Tablo B.8	4. forma ait sonuçlar.	79
Tablo B.9	Şekil B.7’de bulunan karakterlerin ait olduğu mimariler.	81
Tablo B.10	5. forma ait sonuçlar.	81



Çekişmeli Üretici Ağlar ile Sıfırdan İki Boyutlu Karakter Üretimi ve Karakter Renklendirilmesinde PixelSight Bloklarının Kullanımıyla Çeşitlilik ve Kalitenin Artırılması

Volkan Can BACAKSIZ

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Dr. Öğr. Üyesi Oğuz ALTUN

Yaptığımız tez çalışmasında, iki boyutlu karakter üretimini otomatikleştirmeye odaklanan çalışmaları gözden geçirdik. Henüz bu alanda kullanılmamasına rağmen potansiyel olarak uygulanabilir olabilecek çalışmaları da tartıştık. İki boyutlu karakter üretimin sürecinde kullanılan mimarilere yönelik iyileştirmeler sunduk.

Konunun daha iyi anlaşılmasını sağlamak için, iki boyutlu karakterleri ve bunların üretim süreçlerini tanıttık. 2B karakterlerin üretim süreçleri, özellikle ÇÜA mimarilerinin ortaya çıkmasıyla birlikte önemli ölçüde gelişmiştir. ÇÜA ve VOK tabanlı mimarilere odaklanarak iki boyutlu karakterler için kullanılan temel mimarileri tartıştık. Ayrıca, gelecekteki çalışmalarda kullanılabilecek mimarilerden bahsediyoruz ve bahsettiğimiz mimarilerin neden uygun seçenekler olabileceğini kısaca açıkladık.

Karakter renklendirme için MANGAN mimarisini ve Pix2Pix mimarisinin çeşitli modifikasyonlarını karşılaştırdık. Karşılaştırdığımız Pix2Pix modifikasyonları arasında Isola vd. tarafından önerilen orijinal Pix2Pix mimarisi, Serpa ve Rodrigues tarafından önerilen Pix2Pix mimarisi, Jiang ve Sweetser tarafından önerilen Pix2Pix mimarisi, Coutinho ve Chaimowicz tarafından önerilen Pix2Pix mimarisi bulunmaktadır. Renklendirme için yaptığımız karşılaştırmada, en iyi sonuçları Isola vd. tarafından önerilen orijinal Pix2Pix mimarisi ile elde ettik. Bu nedenle, geliştirmelerimizi Isola vd. tarafından önerilen mimariyi temel alarak

gerçekleřtirdik. Pixel VQ-VAE mimarisindeki PixelSight bloklarından esinlenerek, üretetin ilk katmanına ve ayırıcının son katmanına 1x1 konvolüsyonlar ekledik. Yaptığımız eklemeler sonucunda görüntünün bir bütün olarak renklendirilmesini önledik. Ayrıca, eğitim setindeki renk baskınlığının etkisini HSV histogramı yardımıyla ortaya koyduk. Gerçek dünya verileri ile deneyler yaparak sonuçlarını paylařtık.

Karakter üretimi için DKÇÜA mimarisini kullandık. DKÇÜA mimarisi ile karakter üretiminde sapmalarla karşılařtık. Sapmaların önüne geçebilmek amacıyla tarihsel ortalama tekniğinden esinlenen bir yöntem kullanarak eğitim sırasında sapmaları önledik. Pixel VQ-VAE mimarisindeki PixelSight bloklarından esinlenerek, DKÇÜA mimarisinin ayırıcı ağına 1x1 konvolüsyon ekledik. Çeřitliliğı ve çıktı kalitesini artırdık. Değerlendirme yöntemlerinin karakter üretimini değerlendirmekte yetersiz kaldığına değindik. Düşüncemiz doğrutusunda kullanıcı anketi ile karakter üretimine ait deneyimizi tekrar değerlendirdik.

Anahtar Kelimeler: Karakter üretim süreci otomasyonu, piksel sanatı, karakter üretimi, ÇÜA, VOK.

ABSTRACT

Enhancing Diversity and Quality in Two-Dimensional Character Generation and Character Coloring using Generative Adversarial Networks with PixelSight Blocks

Volkan Can BACAŞIZ

Department of Computer Engineering
Master of Science Thesis

Supervisor: Asst. Prof. Oğuz ALTUN

In this thesis, we reviewed studies focused on automating the generation of two-dimensional characters. To enhance understanding of the topic, we introduced two-dimensional characters and their production processes. We discussed the core architectures used for two-dimensional characters, focusing on GAN and VAE-based models. Additionally, we mentioned architectures that could be utilized in future research and briefly explained why these architectures may be suitable options.

We compared the MANGAN architecture and various modifications of the Pix2Pix architecture for character coloring. In our comparison, the best results were achieved with the Pix2Pix architecture proposed by Isola et al., which was the original Pix2Pix architecture. Therefore, we based our improvements on this architecture. Inspired by the PixelSight blocks in the Pixel VQ-VAE architecture, we added 1x1 convolutions to the first layer of the generator and the last layer of the discriminator. This enhancement led to progress in terms of similarity and preventing the image from being colored as a whole. We used an HSV histogram to analyze color dominance in the training set and showed its impact on results. Also, we experimented with real-world data.

We used the DCGAN architecture for character generation. During character generation with the DCGAN architecture, we encountered deviations. To prevent

these deviations, we implemented a method inspired by the historical averaging technique, which helped to mitigate deviations during training. Inspired by the PixelSight blocks from the Pixel VQ-VAE architecture, we incorporated a 1x1 convolution into the discriminator network of the DCGAN architecture. This modification enhanced diversity and output quality. We also addressed the insufficiency of current evaluation methods in assessing character generation. Based on this, we reassessed our character generation experiment using a user survey.

Keywords: Character generation process automation, pixel art, character generation, GAN, VAE.



Karakter üretim süreci ile ilgili önemli bilgileri açıklayıcı bir şekilde ele aldık. Karakter üretim sürecinde karşılaşılabilecek terimleri ve aşamaları detaylandırdık. Literatürde genellikle oyunlarla ilgili çalışmalar bulunması nedeniyle, açıklamalarımızı oyun süreçleri üzerinden yürüttük.

1.1 2 Boyutlu (2B) Çizim

2 boyutlu (2 dimensional) çizimler birçok farklı alanda kullanılan bir tasarım biçimidir. Teknolojinin gelişmesiyle beraber farklı tasarım biçimleri ortaya çıkmıştır. 3 boyutlu (3B) çizimler 2B çizimlere kıyasla daha fazla kullanılır hale gelmiştir. Ancak 2B çizimler hala geçerliliğini korumaktadır. Özellikle 2B çizimler kullanılarak ortaya konan oyunlar ve çizgi romanlar popülerliğini korumaktadır.

Oyunlarda gerçekçi 3B (3 dimensional) çizimlere sahip olmak her zaman iyi olmayabilir. Çünkü oyunlarda hayal gücüne dayalı bir üretim bulunmaktadır. Yapılmak ve kullanıcıya aktarılmak istenen deneyim, bazen 2B çizimleri gerektirebileceği gibi, 2B çizimler maliyet açısından da avantajlı olmasından kaynaklı kullanılabilir.

2B çizimler 3B çizimler için referans olarak da kullanılabilir. 3B bir karakter çizilirken 2B olarak çizilen bir karakterin yan görüntüsü, 3B karakteri oluştururken, karakterin derinliğini oluşturmada yardımcı olacaktır. 3B çizim için özellikle oyunlarda referans olarak 2B çizimlerin kullanılması yaygın bir yöntemdir.

1.1.1 Piksel Sanatı

Oyun veya çizgi romanların 2B olarak ortaya konması durumunda 2B karakterler kullanılır (Şekil 1.1.a). 2B karakterler üretilirken farklı sanat tarzları kullanılabilir. Piksel sanatı (pixel art) da bu sanat tarzlarından biridir (Şekil 1.1.b). Piksel sanatı dijital dünya özelinde kullanılır. Piksel sanatı başlangıçta

sınırlı grafik özellikleri için tercih edilmiştir [1]. Sınırlı grafik özelliklerine çok iyi uyum sağladığından o dönemlerde daha yaygın kullanılmıştır. Gelişen teknolojiyle beraber piksel sanatı da gelişmiştir. Başlangıçtaki kadar yaygın olmasa da daha erişilebilir ve maliyet açısından daha uygulanabilir olmasından dolayı bağımsız oyun üreticileri tarafından kullanılması, ve retro oyun stili severlerin tercih etmesi nedeniyle hala yaygın bir sanat tarzıdır.



Şekil 1.1 (a) 2B karakter yüzü [2]. (b) Piksel sanatı karakter [3].

Dijital bir ortamda görüntünün temel birimi piksel (pixel) olarak adlandırılır [4]. Bir piksel kabaca kare bir bloktur. Kare bloklar birleşerek bir görüntüyü oluşturur. Piksellerin birleşmesi ile oluşturulan bu görüntü piksel sanatı olarak adlandırılır. Piksel sanatı oluşturulurken sınırlı sayıda piksel kullanıldığı için yoğun bir emek ve çaba ister [1, 5]. Oldukça hassas ve kusursuz bir yapıya sahiptir [6]. Ekrandaki her bir pikselin kasıtlı olarak yerleştirilmesi gerekir [7]. Bu süreç zamana mal olmaktadır. Piksel sanatı, zaman ve beceri açısından belirli gereksinimler taşısa da, maliyet açısından tasarruf sağlar. Kısaca piksel sanatı dijital ortamın en küçük görüntü birimi olan pikselleri sınırlı sayıda kullanarak 2B görüntü ve animasyon oluşturulmasını sağlayan dijital bir sanattır [8].

Piksel sanatı teknolojinin gelişmesiyle beraber gelişmiştir. Geçmişte 8 bit (byte) ve 16 bit gibi sınırlandırmalarla karşı karşıya olan piksel sanatı bu kısıtlamalardan kurtulmuştur [8]. Yaklaşık olarak 16 milyon farklı renk ile renklendirilebilir [4]. Dolayısıyla kullanıcıya aktarılacak istenen görüntüde bir kayıp olmamaktadır. Bu gelişmeler bağımsız oyun yapımcılarının bu tarzı kullanmasına olanak sağlar. Çünkü çağın gerisinde kalmış bir sanat türü değildir. Aksine piksel sanatı temel bir görüntünün üzerine hayal kurulabilmesini sağlar. Gücünü bu etkisinden alır.

1.2 Karakter Üretim Süreci

Genel olarak bakıldığında oyun yapımı uzun ve zorlu aşamaları bulunan bir süreçtir [9]. Bu süreçlerden biri de tasarım aşamasıdır. Çünkü oyunlar içerisinde iletişim halinde olan binlerce farklı sanat varlığı (asset) bulunabilir [10, 11]. Gelişen teknoloji ile beraber tasarım aşamasındaki maliyet de önemli ölçüde artmıştır [12]. Oyunlar oyun türleri, konsept ve oyun geliştirici ekibin bütçesi gibi nedenlerle farklı tasarım tarzlarına yönelebilmektedir [13]. Bu tarzlardan biri olan 2 boyutlu çizimler beceri ve zaman gerektirirken maliyetten tasarruf sağlar [14]. Bu nedenle bağımsız oyun geliştiricilerin gözdesidir. Günümüz dünyasında piksel sanatı genel olarak retro oyunlar için kullanılır. Aynı zamanda bağımsız oyunlar tarafından da daha az maliyetle stil sahibi oyunlar ortaya çıkarmak için kullanılır.

1.2.1 Karakter Üretimi

Oyunlarda kullanıcılarla etkileşimi sağlayan ana unsur ekrandaki görüntüdür [15, 16]. Ekrandaki görüntüyle beraber tema ve hissiyatın aktarılması önemlidir [17]. Hissiyat olarak bakıldığında karakter üretimi oldukça önemlidir. İnsanlar ekrandaki karakterlere kişilik atamaya meyillidir. Oyunlarda da durum tam olarak böyledir. Özellikle oyunda ana karakter bulunuyorsa, ana karakter tüm oyun deneyimine etki edecektir [18]. Çünkü oyuncu doğrudan ana karakter ile etkileşim içerisinde. Oyuncunun oyunla veya ana karakterle bağ kurabilmesi oyunun tekrar oynanabilirliğini güçlendirecektir.

Ana karakterin yanı sıra oyunda bulunan diğer karakterlerin de önemi azımsanmayacak seviyededir. Düşman karakterlerin tasarımları, oyuncu olmayan karakterlerin (Non-player character (NPC)) tasarımları özenle gerçekleştirilir. Oyndaki karakter çeşitliliği fazla olabileceği gibi, az sayıda da olabilir. Özellikle çeşitli karakterlere sahip oyunlarda otomasyon daha ön plana çıkacaktır.

Karakterler taşıdıkları özelliklerle uyum içerisinde olan bir görünüme sahip olmalıdır. Bu durum bazı oyunlar için geçerli olmasa da çoğunlukla geçerliliğini korumaktadır. Atik olan bir oyun karakterinin daha zayıf olması beklenirken, güçlü ancak çeviklik konusunda sıkıntıları olan bir oyun karakterinin daha yapılı olması beklenir. Yapılan karakter tiplmesi ile bağdaşan özellikler kullanıcının tezatlık yaşamaması için önemlidir.

1.2.2 Karakter Renklendirmesi

Herhangi bir karaktere ait renklendirme işlemi yapılırken dikkat edilmesi gereken bazı unsurlar bulunmaktadır. Oyunun teması, arka planla bir uyum

içerisinde olurken aynı zamanda ayırt edilebiliyor olması ve karakter üzerinde kullanılan renklerin birbiriyle uyumlu olması gibi unsurlar dikkat edilmesi gereken unsurlardan bazılarıdır. Kötü bir karakter renklendirmesi, iyi bir karakteri bile kötü gösterebilir.

Renklendirme için farklı yöntemler bulunmaktadır. Tek renk tonunu kullanarak renklendirme sağlanabileceği gibi, farklı renklerle de renklendirme yapılabilir. Farklı renklendirme çeşitlerini gözlemleme süresini kısaltmak tasarımcıya ilham olabileceği için karakter renklendirmesi ile alakalı otomasyon çalışmaları önem taşımaktadır.

Sadece renklendirme üzerine olan çalışmalarda tasarımcı otomasyon ile üretilen renklendirilmiş karakterleri kullanmak zorunda değildir. Kullanılabilir olması ilk önceliklerden biri olsa da, tasarımcıya ilham kaynağı olması da büyük bir önem taşımaktadır. Otomasyon tarafından kullanılan renk paleti beğenilirse, karakter otomasyon tarafından kullanılan renk paleti ile tasarımcı tarafından renklendirilebilir.

1.2.3 Karaktere Ait Animasyon Oluşturma

Karakter üretiminin yanı sıra, karaktere ait animasyon üretimi de zor ve yoğun bir süreçtir [19]. Özellikle 2B hareketli çizim (sprite) üretimi, zorluğundan dolayı oyun geliştirme süreci için büyük bir yük oluşturur [20]. Hareketli çizim bir karakterin veya bir nesnenin animasyonunu oluşturmak için kullanılan bir yöntemdir. Animasyon için ekrana yansması gereken görüntüler tek tek çizilir ve art arda oynatılır (Şekil 1.2). 2B hareketli çizim üretim süreci tüm oyun tasarım sürecinde olduğu gibi referans toplanmasını da gerektiren bir arama süreciyle başlar [21]. Animasyonu yapılacak hareketin her bir karesinin ölçülü biçimde çizilmesi gerekir. Çizilen kareler belirli aralıklarla oynatılarak akıcılık sağlanır. Aynı zamanda animasyonlar döngü olarak da oynatılabilir. Animasyonlar arasında geçiş yumuşak olmalı, fark edilmemelidir.



Şekil 1.2 Oyun karakteri hareketli çizimi [3]. Şekil 1.1.b'de gösterilen karaktere ait hareketli çizimdir.

Animasyonlar etkileşimi sağlayan kısımlara sahip olabileceği için uzunlukları da

önemlidir. Saldırı animasyonunun uzun olması, oyun içerisinde bulunan diğer bir sanat varlığına vurulduğunda oluşacak etkinin, animasyon tamamlanmadan önce ortaya çıkmasına neden olur. Henüz saldırı animasyonu bitmemiş iken, oluşacak etki görülmüş olur. Kullanıcıya aktarılan hissiyatın büyük oranda kırılmasına neden olan bu durum, oyun motorları (game engine) içerisinde çözülebilir. Ancak dikkat edilmesi gereken bir husustur.

1.3 Karakter Üretiminin Otomasyonlaştırılması

Tanıtığımız ve zorluklarından bahsettiğimiz karakter üretim sürecinin otomasyonlaştırılması oldukça önemlidir. Karakter üretim süreci ile ilgili otomasyon geliştirmek tasarımcının iş yükünü azaltacaktır. Aynı zamanda tasarımcıya ilham kaynağı olabilme potansiyelini taşımaktadır. Otomasyonun gerçekleştirilmesi sürece ayrılan maliyetin ve zamanın azalmasını sağlayacaktır. Sağlanan avantaj sayesinde geliştirilen projelerin süresinde ve kalitesinde iyileştirme sağlanabilir.

2B karakter sürecini otomatikleştirmek isteyen çalışmalar genel olarak görüntüden görüntüye (image to image (I2I)) mimarilere yönelmişlerdir. Görüntüden görüntüye mimariler girdi (input) olarak görüntü isteyen ve çıktı (output) olarak da görüntü veren mimarilerdir. Bir görüntüden yeni bir görüntüyü elde etmek için kullanılabilirler. Aynı zamanda varyasyon üretimi için de görüntüden görüntüye mimariler yaygın olarak kullanılmaktadır.

Bu tez çalışmasında, konunun daha iyi anlaşılmasını sağlamak için 2B karakterleri ve 2B karakterlerin üretim sürecini tanıttık. Karakterlerin üretimi ve süreçleri, görüntü işlemedeki ilerlemelere benzer şekilde, özellikle çekişmeli üretici ağ (ÇÜA) mimarisinin ortaya çıkmasıyla birlikte gelişmiştir. Çekişmeli üretici ağ ve varyasyonel otomatik kodlayıcı (VOK) tabanlı çalışmalara odaklanarak 2B karakterler için kullanılan temel mimarileri tartıştık. Ek olarak, gelecekteki çalışmalarda kullanılabilecek mimarilerden bahsettik ve önerdiğimiz mimarilerin neden uygulanabilir seçenekler olabileceğini kısaca açıkladık. Bu şekilde, mevcut kaynakları bir araya getirdik ve karakter üretim süreci alanında gelecekte yapılacak çalışmalar için mevcut çalışmalar hakkında temel bilgiler sağladık. Araştırmamızda çekişmeli üretici ağ mimarilerine daha fazla odaklandık. Yapacağımız iyileştirmelerde piksel vektör kuantize edilmiş varyasyonel otomatik kodlayıcı (Pixel VQ-VAE) mimarisinde yapılan yeniliklerden ilham aldığımız için varyasyonel otomatik kodlayıcı mimarilerine (variational autoencoder) de değindik. Aynı zamanda varyasyonel otomatik kodlayıcı mimarilerinin de karakter üretim

sürecinde kullanıldığı çalışmalardan bahsettik.

Karakter boyama süreci için özellikle Pix2Pix alanında yapılan çalışmaları kıyasladık. Karakter üretim süreci için ise derin konvolüsyonel çekişmeli üretici ağ (DKÇÜA) mimarisinde sonuçlar elde ettik.

Renklendirme için yaptığımız kıyaslamada en iyi sonucu veren Pix2Pix mimarisi üzerinden ilerledik ve renklendirme sırasında görüntünün bir bütün olarak renklendirilmesini engelledik. Aynı zamanda orijinale yakın görüntü üretimi açısından da daha iyi sonuçlar elde ettik. Ek olarak, eğitim setindeki renk baskınlığının sonuçlar üzerindeki etkisini gösterdik. Gerçek dünya verilerinin renklendirilmesine dair deneyler yaptık. Gerçek dünya verilerinin renklendirilmesini iyileştirmek amacıyla yeni çizgi sanatı görüntüsü oluşturma yöntemi önerdik. Karakter üretim sürecinde ise, eğitim sırasında meydana gelen sapmaları önledik. Çıktı kalitesini ve çeşitliliği artırdık. Değerlendirme yöntemine olan ihtiyacı detaylandırdık. Kullanıcılar aracılığı ile çıktıları değerlendirdik.

2 YÖNTEMLER

2B karakter üretimi için farklı alanlarda çalışmalar yapılmıştır. Özellikle sıfırdan karakter üretimi ve animasyon hareketli çizimi üretimi konusunda oyunlarla ilişkilendirilmiş makale sayısı oldukça fazladır. Oyunları geliştirme sürecinde yardımcı olması amacıyla farklı yazılımlar kullanılır. Bunlardan biri de prosedürel içerik üretimi (PİÜ) olarak tanımlanmıştır [22]. Prosedürel içerik üretimi (procedural content generation (PCG)), içerik üretme işlemlerinin otomatikleştirilmesidir [23]. Oyunlar için bakıldığında oyun içeriğini otomatikleştiren yazılımlar olarak yorumlanabilir. Makine öğrenmesi, prosedürel içerik üretimi yazılımlarında yaygın olarak kullanılmaktadır. Makine öğrenmesi ile prosedürel içerik üretimi yazılımlarının beraber kullanılması, makine öğrenimi ile prosedürel içerik üretimi (MÖPİÜ) olarak da adlandırılmaktadır [24–26]. Makine öğrenimi ile prosedürel içerik üretimi (procedural content generation via machine learning (PCGML)) yazılımlarında farklı derin öğrenme yöntemleri kullanılabilir. Prosedürel içerik üretimi ve makine öğrenimi ile prosedürel içerik üretimi kavramları ile bu konuda araştırma yapmak, araştırmanın derinliğini artıracaktır.

Prosedürel içerik üretimi yazılımlarının kullanılmasının çoğunlukla asıl nedeni el emeğinin azaltılmasıdır [27]. El emeğinin azaltılmasına bağlı olarak zaman ve maliyetten tasarruf sağlanır. Bu nedenle son yıllarda prosedürel içerik üretimi kullanımının popülaritesi artmaktadır. Bunun yanı sıra prosedürel içerik üretimi yazılımları tasarımcıların yaratıcılıklarını artırmaktadır. Çünkü insanlar birbirini taklit etme eğilimindedir. Prosedürel içerik üretimi yazılımları tam olarak kullanıma uygun bir sonuç üretmese de, tasarımcıya bir ilham kaynağı sunabilir. Prosedürel içerik üretimi yazılımları oyunlardaki bölüm tasarlama, oynanış ve karakter tasarımında kullanılabilir [28].

2B karakterlerin üretim sürecine yardımcı olabilmek amacıyla farklı mimariler benimsenmiştir. Amacımız bu mimarileri açıklamak, yapılan çalışmalardan

bahsetmek, bu konuda kullanılabileceğini düşündüğümüz ancak henüz kullanılmamış mimarilere değinmektedir.

2.1 Optimizasyon Yöntemleri

Mimarileri açıklama kısmına girmeden hemen önce bu mimarilerde yaygın olarak kullanılan stokastik gradyan iniş (SGİ) ve Adam optimizasyon yöntemlerini açıkladık. Stokastik gradyan iniş (stochastic gradient descent (SGD)) yöntemi derin öğrenmede çokça kullanılan bir optimizasyon yöntemidir. Stokastik gradyan iniş optimizasyon yöntemi her bir eğitim seti için güncelleme yapar. Bu sayede yerel minimuma daha erken sürede ulaşması beklenir.

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} L(\theta_t; x_t, y_t) \quad (2.1)$$

Stokastik gradyan iniş yöntemi, Denklem 2.1’de gösterilmiştir [29, 30]. θ_{t+1} hesaplanacak olan parametrelerdir. θ_t güncel parametrelerdir. α öğrenme katsayısını temsil eder. Öğrenme katsayısı sayesinde öğrenme işleminin hızı ayarlanabilir. $\nabla_{\theta} L(\theta_t; x_t, y_t)$ ifadesi ise rastgele seçilen örneklerle göre kayıp fonksiyonunun gradyanını temsil eder. $(\theta_t; x_t, y_t)$ ifadesinde bulunan x_t girdiyi (input), y_t ise etiketi (label) belirtir. Stokastik gradyan iniş yönteminde öğrenme katsayısı sabit tutulur.

Daha hızlı yakınsaması ile bilinen Adam yöntemi, birden çok denklem ile tanımlanmaktadır:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_{t+1} \quad (2.2)$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) g_{t+1}^2 \quad (2.3)$$

$$\theta_{t+1} = \theta_t - \alpha_{t+1} \frac{m_{t+1}}{\sqrt{v_{t+1}} + \epsilon} \quad (2.4)$$

m_{t+1} birinci momentumu gösterir (Denklem 2.2) [31, 32]. Momentum hesaplanırken daha önce hesaplanan gradyanlar dikkate alınır. β_1 bir hiper parametredir (hyperparameter). m_t bir önceki momentum değerini gösterir ve β_1 parametresi önceki momentumun önemini ayarlar. g_{t+1} ise gradyanı gösterir.

Denklem 2.3'te bulunan v_{t+1} ise ikinci momentumu ifade eder. İkinci momentum hesaplanırken β_1 'e benzer bir parametre olan β_2 kullanılır. Gradyanın karesi ve bir adım önceki ikinci momentum dikkate alınarak hesaplama yapılır. Gradyanın karesini almak aslında gradyanın kendisi ile Hadamard çarpımına denk gelir ($g_t^2 = g_t \odot g_t$). Hiper parametreler için sıklıkla $\beta_1 = 0.9$, $\beta_2 = 0.999$ ve $\epsilon = 10^{-8}$ değerleri kullanılır.

Adam optimizasyon yöntemi, stokastik gradyan iniş kiyasla daha kötü genelleme yapmasına rağmen daha hızlı yakınsaması nedeniyle tercih edilmektedir. Stokastik gradyan iniş yöntemi gibi, Adam yönteminin de uygulaması kolaydır, verimlidir ve düşük bellek gerektirir. Adam yöntemi bu özellikleri sayesinde gürültülü gradyanlarla başa çıkmak için uygun hale gelir [33].

Genelleme konusunda stokastik gradyan iniş yöntemi daha iyi çalışmaktadır. Ancak çekişmeli üretici ağ ve pekiştirmeli öğrenme (reinforcement learning) mimarileri için Adam yönteminin genellemeyi kötü yapması göz ardı edilebilir. Aynı zamanda Adam optimizasyon yönteminin çok tercih edilmesinin bir diğer nedeni de ayarlanabilirlik ölçeğinin yüksek olmasıdır.

2.2 Çekişmeli Üretici Ağ (ÇÜA) Mimarileri

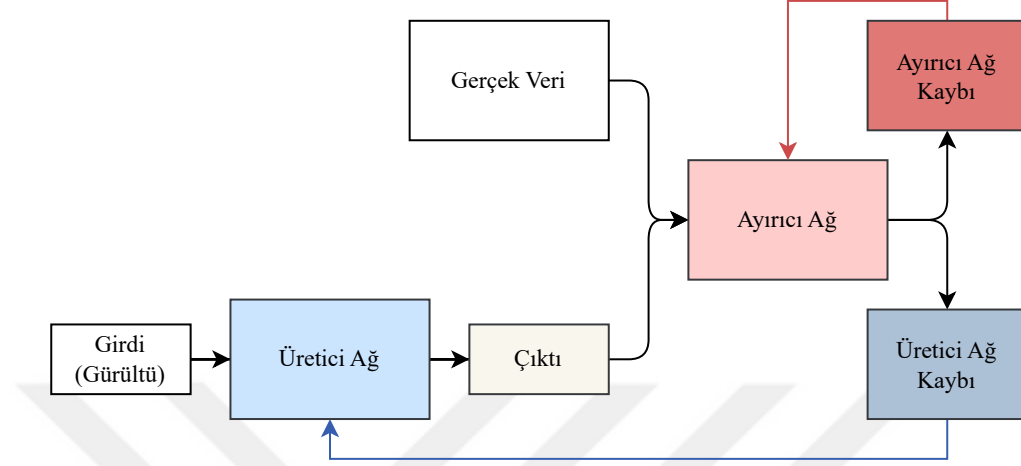
İlk olarak klasik üretken çekişmeli ağları tanıttık ve ardından özellikle problemimizle ilgili çekişmeli üretici ağ versiyonlarını gözden geçirdik.

2.2.1 Klasik Çekişmeli Üretici Ağ (ÇÜA)

Ian Goodfellow vd. [34–36] tarafından ortaya atılan bir mimaridir. Popüler olarak gürültüden kullanılabilir bir veri elde edilmek için kullanılır. Elde edilecek veri, görüntü veya vektör gibi farklı çeşitlerde olabilir. Çekişmeli üretici ağ (generative adversarial network (GAN)) mimarisi temel alınarak yapılan çalışmalarla beraber görüntüden görüntü üretme çalışmalarında önemli gelişmeler sağlanmıştır [37].

Çekişmeli üretici ağ mimarisi 2 farklı yapay sinir ağı içerir (Şekil 2.1). Bu sinir ağlarından biri üretici ağ (generator) olarak adlandırılır. Üretici ağı girdi olarak verilen görüntü veya gürültüden yeni görüntüler üretir. Yapılmak istenen işleme göre üretilen görüntü farklı çekişmeli üretici ağ mimarileriyle beraber yönlendirilebilir. Çekişmeli üretici ağ mimarisindeki diğer sinir ağı ise ayırıcı ağ (discriminator) olarak adlandırılır. Bu ağın görevi ise üretici tarafından üretilen görüntüyü gerçek veya sahte olarak yorumlamaktır. Olasılığa dayalı olarak ayırım yapılır. Yapılan bu işlem sonucunda yorum üretici ağın eğitiminde kullanılır.

Dolayısıyla ayırıcı ağın çıktısı (output) ile eğitilen üretici ağ bir sonraki çıktısını günceller ve kendisini geliştirir. Bu şekilde hem üretici ağ hem de ayırıcı ağ birbiriyle çekişerek gelişim sağlar. Çekişmeli üretici ağ mimarisi bu temel üzerine kuruludur.



Şekil 2.1 Çekişmeli üretici ağ mimarisine genel bakış.

Çekişmeli üretici ağ mimarisinin kurulu olduğu temel aslında sıfır toplamlı oyun (zero-sum game) olarak bilinen bir oyun teorisi (game theory) kavramına dayanmaktadır [34]. Üretici ağ, ayırıcı ağı yanıltmak için daha gerçekçi görüntüler üretirken, ayırıcı ağ ise görüntüleri daha iyi ayırt etmeye çalışır. Sıfır toplamlı oyun kuramı bu rekabeti içerir. Sıfır toplamlı oyun teorisinde, bir katılımcının kazançları diğer bir katılımcının kayıpları ile dengelenir.

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2.5)$$

Çekişmeli üretici ağ mimarisi Denklem 2.5'te gösterilen maliyet fonksiyonu (objective function) üzerinden eğitimini sürdürür. Denklemde G (generator) üretici ağ, D (discriminator) ise ayırıcı ağı temsil eder. x gerçek verileri belirtir. z gürültüyü belirtmektedir. $\mathbb{E}_{x \sim p_{\text{data}}(x)}$ ifadesi gerçek veriler içerisinde rastgele bir x değerinin beklenen değerini gösterir. $\mathbb{E}_{z \sim p_z(z)}$ ifadesi rastgele bir gürültünün beklenen değerini sembolize etmektedir. $D(x)$ ayırıcı ağın gerçek veri için verdiği çıktıyı gösterir. Ayırıcı ağ gerçek veriyi doğru bir şekilde ayırt etmeyi başardıkça bu değer 1'e yaklaşır. $\log(1 - D(G(z)))$ ifadesindeki $G(z)$ değeri üreticinin z girdisi ile ürettiği çıktıyı temsil eder. $D(G(z))$ ifadesi ile beraber üreticinin ürettiği çıktının ayırıcı ağa verilmesi sonucu elde edilen çıktılar ayırıcı ağ tarafından ayırt edilmeye çalışılır. Ayırıcı ağ sahte veriyi doğru ayırt ettikçe $D(G(z))$ 0

değerine yaklaşıacaktır. $1 - D(G(z))$ değeriyle beraber ayırıcı ağın sahte veriyi ayırt edemediği durumlar elde edilir. Logaritma kullanarak hesaplama daha hassas hale getirilir.

Üretici ağ, θ_g parametrelerini ayarlayarak $V(D, G)$ değer fonksiyonunu en aza indirmeyi amaçlar. Ayırıcı ağ, θ_d parametrelerini ayarlayarak aynı değer fonksiyonunu $V(D, G)$ maksimize etmeyi amaçlar.

Başka bir deyişle, D ve G , V ile iki oyunculu bir sıfır toplamli oyun oynar [34]. Ian Goodfellow vd. [34], oyunu yinelemeli bir sayısal yaklaşım kullanarak uygulamayı önermektedir: iç döngüde, G 'yi sabit tutarak k adımları için D 'yi optimize ederler ve dış döngüde, D 'yi sabit tutarak G 'yi bir adım için optimize ederler.

Ayrıca, ayırıcı ağın sabit tutulması sonucu, üretici ağın optimizasyon problemi şu hale gelir:

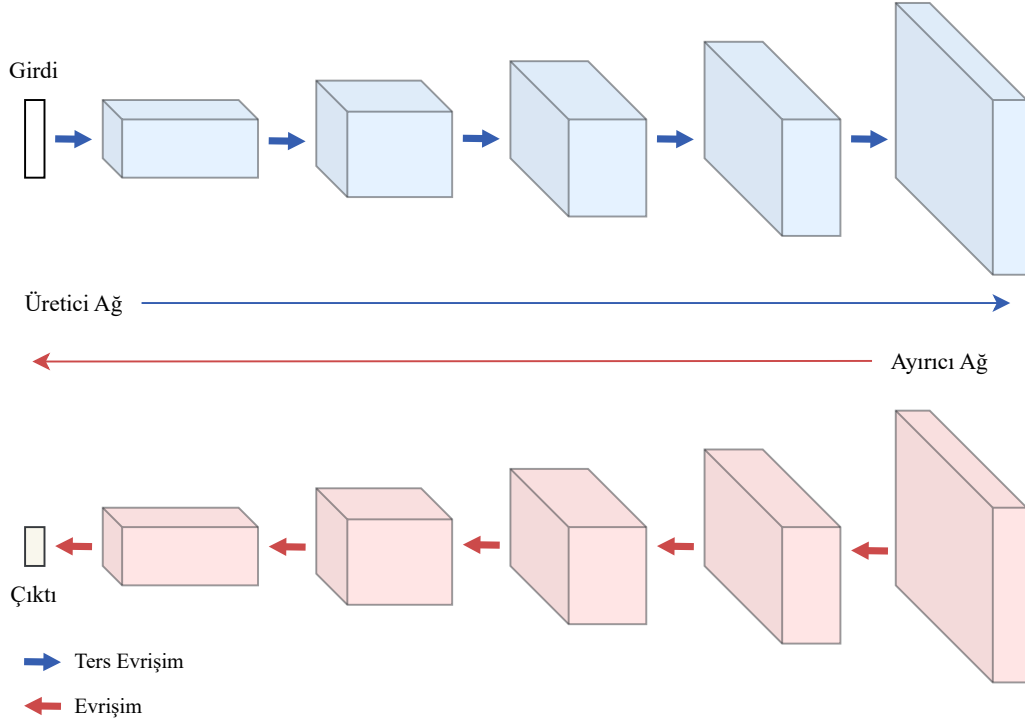
$$\min_{\theta_g} \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z; \theta_g)))] \quad (2.6)$$

Yazarlar ayrıca erken öğrenme aşamalarında $\log(1 - D(G(z)))$ değerini minimize etmek yerine $\log D(G(z))$ değerini maksimize etmeyi önermektedir. Düşük performanslı bir G ile D sahte görüntüleri kolayca reddedebilir ve $\log(1 - D(G(z)))$ 'nin doygunluğa ulaşmasına ve zayıf gradyanlar sağlamasına neden olabilir.

Klasik çekişmeli üretici ağ mimarisi genellikle stokastik gradyan inişi ile optimize edilir.

2.2.2 Derin Konvolüsyonel Çekişmeli Üretici Ağ (DKÇÜA)

Çekişmeli üretici ağ mimarisini baz alarak ortaya çıkmış bir mimaridir. Derin konvolüsyonel çekişmeli üretici ağ (deep convolutional generative adversarial network (DCGAN)) mimarisi çekişmeli üretici ağ mimarisinin aksine görüntü üzerinde işlem yapmak için daha idealdir [38]. Derin konvolüsyonel çekişmeli üretici ağ mimarisinin, çekişmeli üretici ağ mimarisine göre farkı üretici ağ ve ayırıcı ağ içerisinde evrişimler (convolutions) bulundurmasıdır (Şekil 2.2). Halihazırda görüntü işlemede kullanılan evrişimler sayesinde görüntünün önemli özellikleri elde edilir. Bu özellikler çekişmeli üretici ağ mimarisine entegre edilerek süreç ilerler. Entegre edilen evrişim ağları sayesinde çekişmeli üretici ağ mimarisi daha kararlı bir hale gelir. Üretici içerisinde ters evrişimler bulunurken, ayırıcı ağ içerisinde evrişimler kullanılmaktadır. Bu yapısı sayesinde derin konvolüsyonel



Şekil 2.2 Derin konvolüsyonel çekişmeli üretici ağ mimarisine genel bakış.

çekişmeli üretici ağ mimarisi, görüntü ile alakalı çalışmalarda çekişmeli üretici ağ mimarisine göre daha iyi sonuçlar vermektedir.

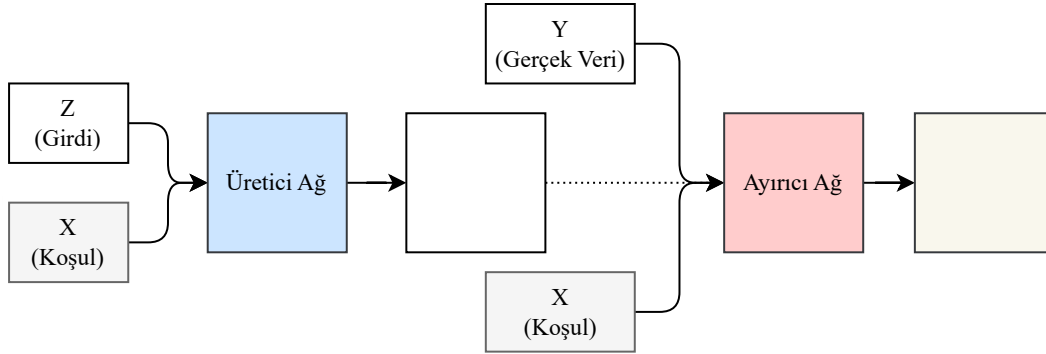
Derin konvolüsyonel çekişmeli üretici ağ mimarisinde yaygın olarak kullanılan bir teknik olan toplu normalizasyon (batch normalization), çıktıların daha hızlı ve daha yüksek kalitede üretilmesine yardımcı olur [39]. Derin konvolüsyonel çekişmeli üretici ağ mimarisinin Adam yöntemiyle optimize edilmesinin daha iyi sonuçlar verdiği gösterilmiştir [40].

Derin konvolüsyonel çekişmeli üretici ağ mimarisi maliyet fonksiyonu olarak klasik çekişmeli üretici ağ mimarisinin kullandığı maliyet fonksiyonunu kullanır (Denklemler 2.5). Temel fark derin konvolüsyonel çekişmeli üretici ağ mimarisi içerisinde derin konvolüsyon ağlarının bulunmasıdır.

Horsley ve Perez-Liebana [41] derin konvolüsyonel çekişmeli üretici ağ mimarisini kullanarak karakter üretimine odaklanmıştır. Derin konvolüsyonel çekişmeli üretici ağ mimarisinin piksel sanatında karakter üretiminde kullanılabileceğini göstermiştir. Yaptıkları çalışmada görüldüğü üzere, derin konvolüsyonel çekişmeli üretici ağ mimarisi, piksel sanatı ile karakter üretiminde başarılı gözükmektedir. Ayrıca, yaptıkları çalışmada optimizasyonu Adam yöntemi ile sağlamışlardır.

2.2.3 Koşullu Çekişmeli Üretici Ağ (KÇÜA)

Koşullu çekişmeli üretici ağ (conditional generative adversarial network (CGAN)) mimarisi verilen koşula göre yönlendirilebilen bir çekişmeli üretici ağ mimarisidir (Şekil 2.3) [42]. Çekişmeli üretici ağ mimarisinde üretilen verilerde, veriler bir koşula bağlanamaz ve yönlendirilemez. Ancak bu mimari ile beraber üretilmek istenen veri belirli bir koşula (condition) bağlanır. Bu sayede veri üretimi yapılırken yönlendirme sağlanmaya çalışılır. Veri üzerinde etki etmesi istenen koşul hem üretici ağ hem de ayırıcı ağ üzerinde etkilidir.



Şekil 2.3 Koşullu çekişmeli üretici ağ mimarisine genel bakış [42].

Koşullu çekişmeli üretici ağ mimarisi girdi olarak çekişmeli üretici ağ mimarisinde olduğu gibi bir gürültü alır. Ancak bu gürültüye ek olarak bu gürültüye uygulanması istenen koşulu da girdi olarak almaktadır. Dolayısıyla koşullu çekişmeli üretici ağ mimarisi ve koşullu çekişmeli üretici ağ mimarisini baz alarak ortaya çıkan mimarilerde gürültünün yanı sıra bir koşul girdisi olmak zorundadır. Verilen koşul sayesinde mimarinin üretmesi istenen çıktıyı anlaması ve bu istenen çıktıya yaklaşacak biçimde sonuçlar üretmesi beklenir.

$$\mathcal{L}_{CGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \quad (2.7)$$

Koşullu çekişmeli üretici ağ mimarisi Denklem 2.7’de belirtilen maliyet fonksiyonunu kullanır. Maliyet fonksiyonu genel olarak klasik çekişmeli üretici ağ mimarisi baz alındığından benzerlik taşımaktadır. Ancak $D(x, y)$ ve $D(x, G(x, z))$ ifadeleri farklılığı oluşturmaktadır. $D(x, y)$ ifadesi ayırıcı ağın gerçek verinin koşula ait olup olmadığını değerlendirdiğini gösterir. $G(x, z)$ ifadesi ile üretici ağın x koşul verisi ile z olarak belirtilen giriş türünü (gürültü veya giriş görüntüsü) girdi olarak aldığı belirtilir. Üretici ağ bu işlem sonucunda z girdisinden bir çıktı üretir ve $D(x, G(x, z))$ ayırıcı ağın, üretici ağ tarafından üretilen çıktıyı koşul ile beraber değerlendirdiğini belirtmektedir. Maliyet fonksiyonunda belirtildiği üzere üretici ağ ürettiği verileri koşula uygun bir biçimde gerçek veriye benzetmeye çalışır.

Dolayısıyla üretilen çıktılar koşullandırılabilir ve yönlendirilebilir.

Coelho vd. [43] MANGAN ismini verdikleri mimariyi koşullu çekişmeli üretici ağ mimarisinden etkilenerek düzenlemişlerdir. MANGAN mimarisinde konvolüsyonel ağları kullanmışlardır. Yaptıkları çalışma ile çizgi sanatını (line art) boyamayı hedeflemişlerdir. Çizgi sanatı görüntülerini ise orijinal görüntülere uyarlanabilir eşik (adaptive threshold) yöntemini uygulayarak elde etmişlerdir. Koşullu çekişmeli üretici ağ mimarisinde olduğu gibi MANGAN mimarisi de bir koşula göre ilerleme sağlamaktadır. Bu nedenle girdi olarak verilen el çiziminin yanında bir koşul girdisi gerekmektedir. Coelho vd. [43] koşul girdisi için orijinal görüntülerden elde ettikleri renk ipucu (color hint) görüntüsü kullanmışlardır. Renk ipucu görüntüsü, orijinal görüntünün renk dağılımını içerir. Mimarilerine girdi olarak hem çizgi sanatı çizimi hem de renk ipucu görüntüsü verdiklerinde, çıktı olarak oldukça başarılı sonuçlar elde etmişlerdir. MANGAN mimarisinde Adam yöntemi ile optimizasyon sağlanmaktadır.

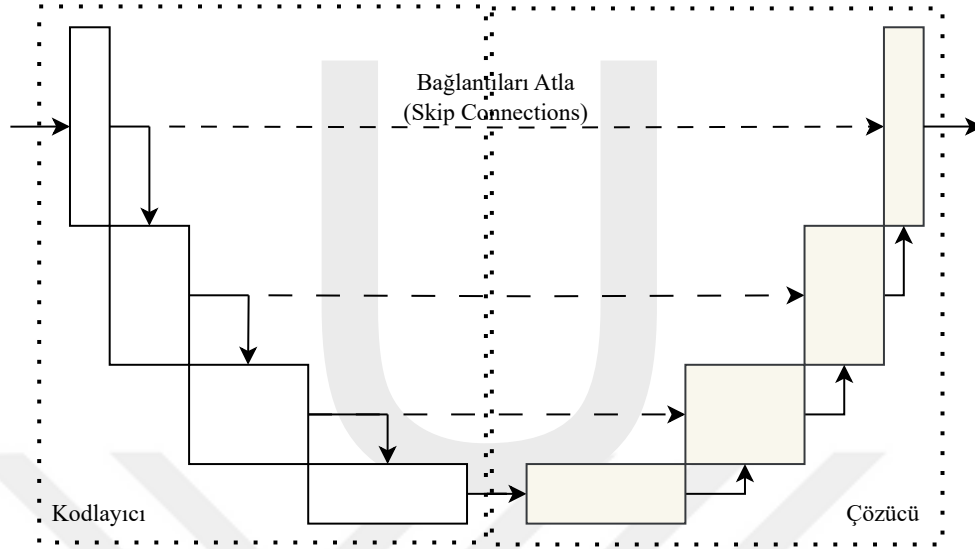
Hui Ren vd. [44] koşullu çekişmeli üretici ağ mimarisini iki aşamalı bir şekilde kullanarak renklendirme yapmayı denemişlerdir. Yaptıkları çalışmada ilk aşamada birkaç özellik noktasında renk bilgisinin öğrenilmesini sağlamışlardır. Bu işlem sonucunda görüntünün gri tonlamalı hali veya piksel düzeyinde ayrılmış (pixel-level parsing) hali ortaya konur. İkinci aşamada ise renklendirme işleminin yapılması sağlanır. Yaptıkları çalışma ile bu alanda yapılan önceki çalışmalara kıyasla ilerleme kaydetmişlerdir. Çalışmada Adam yöntemi ile optimizasyon sağlamışlardır.

2.2.4 Pix2Pix

Pix2Pix mimarisi koşullu çekişmeli üretici ağ mimarisini temel alan bir mimaridir [45]. Pix2Pix mimarisi görüntülerin belirli bir görüntüye çevrilmesi için kullanılır. Pix2Pix mimarisi koşullu çekişmeli üretici ağ mimarisinde de olduğu gibi bir koşula bağlıdır. Dolayısıyla girdinin yanı sıra koşul girdisini de içermelidir. Pix2Pix mimarisi genel hatlarıyla şunları içerir: U-Net üretici ağı, yama tabanlı (patch-based) ayırıcı ağ ve L1 kayıp fonksiyonu (loss function).

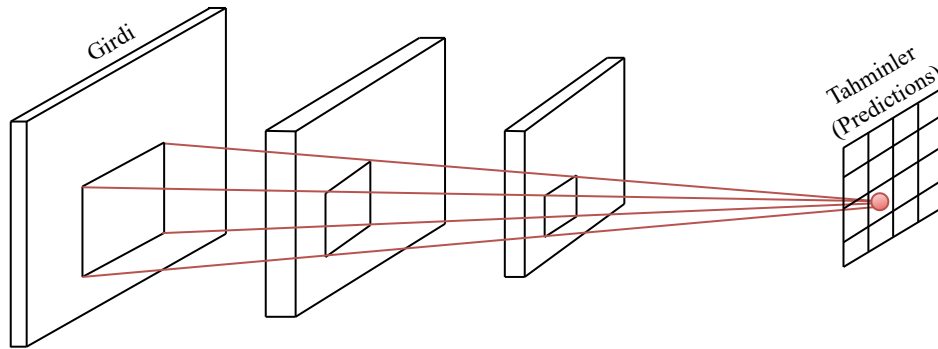
U-Net Üretici Ağı (Şekil 2.4) [46]: Kodlayıcı (encoder) ve çözücüden (decoder) oluşur. Kodlayıcı her bir adımda görüntünün özelliklerini çıkartır ve boyutunu azaltır. Çözücü ise görüntüyü eski boyutuna getirme görevini yapar. Dolayısıyla üretici ağ verilen girdiyi anlar ve üretici ağ için girdiyi çıkış alanına anlamlı bir biçimde çevirmeyi olanaklı kılar. Bu üretici ağ sayesinde görüntünün kullanışlı verileri tutulurken, diğer verileri elenir. Daha kaliteli ve kullanışlı bir görüntü elde

edilir. U-Net üretici ağı başka çekişmeli üretici ağ mimarilerinde de kullanılabilir.



Şekil 2.4 U-Net üretici ağı [46].

Yama Tabanlı Ayırıcı Ağ (Şekil 2.5) [47]: Klasik ayırıcı ağlar görüntüyü bir bütün olarak ele alır. Bu nedenle gerçek veya sahte olarak sınıflandırma yaparken görüntünün tamamını değerlendirir. Ancak yama tabanlı ayırıcı ağ, görüntüyü küçük parçalara böler. Böldüğü bu parçaları ayrı ayrı sınıflandırır. Dolayısıyla eğitim esnasında ayırıcı ağın sahte olan parçalara odaklanması sağlanır.



Şekil 2.5 Yama tabanlı ayırıcı ağ [47]. Görüntüdeki küçük parçalardan gerçek ve sahte tahminleri (predictions) yapılır.

L1 Kayıp Fonksiyonu [48]: Gerçek değerler ile tahmini değerler arasındaki farkın mutlak değerlerinin ortalamasıdır. L1 kayıp fonksiyonunun Pix2Pix mimarisinde tercih edilmesinin nedeni çıktıyı daha az bulanık halde ortaya koymasındır. Denklem

2.8’de bulunan y istenen değeri (gerçek değer) gösterir. $G(x, z)$ ifadesi ile beraber üretici ağın verdiği çıktı temsil edilmektedir. Üretici ağı yönlendirilen x koşul olarak, z ise girdi olarak yorumlanabilir. Yapılan işlem sonucu istenen değer ile üretici ağın verdiği çıktının farkı L1 normu ile hesaplanır.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z} [\|y - G(x, z)\|_1] \quad (2.8)$$

Nihai Pix2Pix maliyet fonksiyonu Denklem 2.9’da gösterilmiştir. L1 kayıp fonksiyonu ile koşullu çekişmeli üretici ağ maliyet fonksiyonu birleştirildiğinde Pix2Pix maliyet fonksiyonunu verir. Denklemdaki λ parametresi L1 kayıp fonksiyonunun önemini ayarlar. Pix2Pix mimarisi optimizasyon için Adam optimizasyon yönteminden yararlanır.

$$\arg \min_G \max_D \mathcal{L}_{CGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (2.9)$$

Serpa ve Rodrigues [49] Pix2Pix mimarisindeki alt örnekleme (downsampling) işlemine ekstra bir konvolüsyon katmanı eklemişlerdir. L1 kayıp fonksiyonu yerine L2 kayıp fonksiyonunu tercih etmişlerdir [48]. Aynı zamanda sızıntılı düzeltilmiş doğrusal birim (Leaky ReLU) aktivasyon fonksiyonu yerine ise üssel doğrusal birim (ELU) aktivasyon fonksiyonunu kullanmışlardır [50, 51]. Yaptıkları bu değişikliklerle beraber animasyon üretimi ve renklendirmesi için önemli sonuçlar elde etmişlerdir. Ancak özellikle renklendirme için sorunların olduğunu belirtmişler ve belirttikleri sorunları göstermişlerdir.

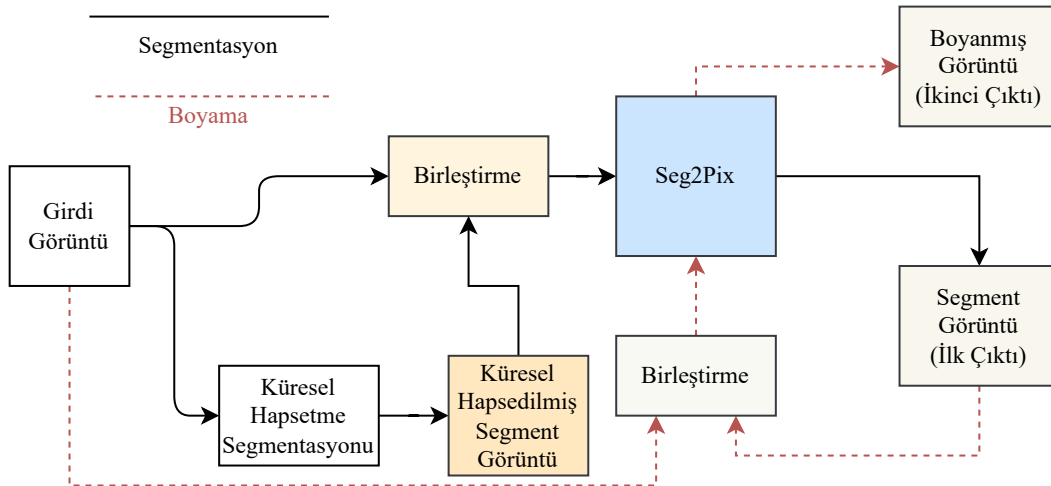
Jiang ve Sweetser [52] Pix2Pix mimarisini YUV renk uzayı (color space) ile birleştirerek karakter renklendirmesi üzerine çalışma yapmıştır. Bu işlemi yaparken öncelikle bir veri seti hazırlamaya çalışmıştır. Bu veri setini hazırlarken orijinal görüntüleri RGB renk uzayından YUV renk uzayına geçirmişlerdir. YUV renk uzayını kullanma nedenleri, bu renk uzayının insan algısına daha yakın bir renk uzayı olması ve iletim hatalarını azaltmasıdır [53, 54]. YUV renk uzayına geçirmiş oldukları görüntünün Y kanalından bir gri görüntü elde edilmiştir. Girdi olarak elde ettikleri bu gri görüntüyü, hedef olarak ise görüntünün orijinalini kullanmışlardır. YUV renk uzayında öğrenme işlemini gerçekleştirmişlerdir. Çalışmada da bahsedildiği üzere girdi ve hedef olarak verilen görüntülerin Y (luminance (parlaklık)) kanalının değerleri eşittir. Bu yüzden mimari sadece U (chrominance blue (mavi renk bileşeni)) ve V (chrominance red (kırmızı renk bileşeni)) kanallarını öğrenmeye çalışacaktır. Bu düşünce ile beraber gerçekleştirdikleri çalışmada karakter renklendirme için başarılı sonuçlar ortaya

koymuşlardır. Stokastik gradyan iniş ve ADAM optimizasyon yöntemleri için farklı çıktılar aldıklarını da belirtmişlerdir.

Coutinho ve Chaimowicz [55, 56] Pix2Pix mimarisinde bulunan toplu normalleştirme yöntemi yerine örnek normalizasyon (instance normalization) yöntemini kullanmayı tercih etmişlerdir. Bu tercihleri ile beraber değiştirdikleri ve çalışmalarında detaylı olarak açıkladıkları mimarilerini kullanarak, karakterin farklı yönlere bakan çizimlerini elde etmek üzerine çalışmışlardır. Girdi olarak bir yöne bakan karakter, koşul yani hedef olarak ise girdi olarak verilen karakterin farklı yöne bakan hallerini vermişlerdir. Yaptıkları başarılı çalışmanın birçok farklı sonucunu paylaşmışlardır.

2.2.5 Seg2Pix

C. W. Seo ve Y. Seo [57] tarafından ortaya atılmış bir mimaridir. Pix2pix mimarisi yerine geliştirdikleri Seg2Pix mimarisini çalışmalarında kullanmışlardır (Şekil 2.6). Seg2Pix için girdi vermeden önce küresel hapsedme (trap ball) segmentasyonu kullanarak girdi olarak kullanılacak görüntüden segment görüntü elde edilmektedir [58]. Daha sonra segment görüntü ve girdi görüntü birleştirilerek mimariye aktarılır. Bu mimaride öz dikkat katmanı (self attention layer) eklenmesinin faydası gözlemlenmektedir. Öz dikkat katmanı görüntüde birbirine uzak olan piksel bağılıklarını, yani görüntünün bütünündeki bütünlüğü korumaktadır. Sıradan konvolüsyon bunu sağlamaz. Dolayısıyla öz dikkat katmanı mimarinin daha iyi bir şekilde çalışmasını sağlar.



Şekil 2.6 Seg2Pix mimarisinin kullanımına genel bakış [57]. Seg2Pix mimarisi 2 farklı işlem için kullanılır. İlk işleminde segment görüntü elde edilir. İkinci işleminde ise istenilen çıktı olan boyanmış görüntü elde edilir.

Öz dikkat katmanı ile beraber girişler 1x1 konvolüsyonlardan geçirilir [59].

Konvolüsyonlar sonucunda sorgu (query ($f(x)$)), anahtar (key ($g(x)$)) ve değer (value ($h(x)$)) matrisleri oluşur. Öncelikle sorgu ve anahtar matrislerinin iç çarpımı ile dikkat haritası (attention map) oluşturulur. Dikkat haritası ile değer matrisinin iç çarpımı sonucu ise öz dikkat haritası (self attention map) oluşturulur ve süreç sonlanır. Öz dikkat katmanı artık sinir ağı (ResNet) bloğundan önce kullanılmaktadır [60]. Karmaşıklık sorunu nedeniyle Seg2Pix mimarisinde sadece üretici ağda kullanılmıştır.

Seg2Pix mimarisi, Pix2Pix varyasyonu bir mimari olduğu için Pix2Pix genel maliyet fonksiyonunu kullanmaktadır (Denklem 2.9). Yapılan çalışmada Adam optimizasyonundan yararlanılmıştır.

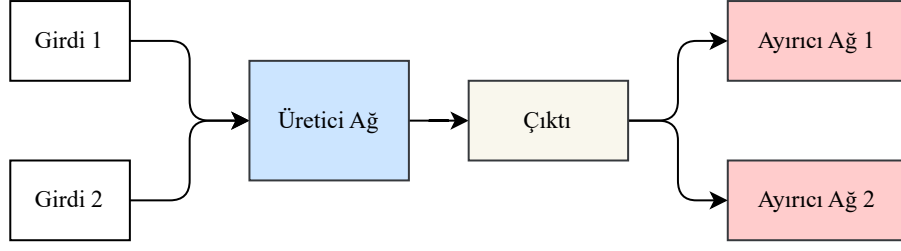
Seg2Pix mimarisi karakter renklendirme işlemini gerçekleştirmek için ortaya atılmış bir mimaridir. Çizgi sanatı bir çizimi renklendirmek amaçlanmıştır. Mimaride girdi olarak çizgi sanatı çizimin yanı sıra küresel hapsetme segmentasyonu yöntemi ile hazırlanmış segment görüntü verilir. Elde edilen segment görüntü çizgi sanatı çizimden elde edilmektedir. Segment görüntü ile çizgi sanatı mimariye verilir. Çıktı olarak yeni bir segment görüntü elde edilir. Başlangıçta girdi olarak verilen çizgi sanatı çizim ile, mimariden çıktı olarak alınan segment görüntü birleştirilir. Birleştirilen bu çizimler mimariye verilir ve son çıktı olarak boyanmış görüntü elde edilir. Seg2Pix mimarisi, özellikle öz dikkat katmanı kullanımı ile beraber, Pix2Pix mimarisine göre daha iyi başarı skoru elde etmiştir.

2.2.6 Çok Ayırıcılı Çekişmeli Üretici Ağ (ÇAÇÜA)

Çok ayırıcılı çekişmeli üretici ağ (multi discriminator generative adversarial network (MDGAN)) mimarisi 2 farklı ayırıcı ağ içeren bir çekişmeli üretici ağ varyasyonudur (Şekil 2.7) [20]. Hong vd. [20] tarafından ortaya konulmuş bir mimaridir. İçerisinde bulunan 2 farklı ayırıcı ağ ile üretilen veriyi 2 farklı yönden inceleyip, farklı yönler üzerinden ayırım yapabilmesiyle öne çıkmaktadır. Bu sayede verilerin daha iyi bir şekilde yönlendirilmesini ve çeşitlendirilmesini sağlar. Aynı zamanda çok ayırıcılı çekişmeli üretici ağ mimarisi, iki farklı ayırmayı yapan ayırıcı ağ verisiyle eğitilen bir üretici ağa sahip olduğundan, üretici ağ eşlemeyi öğrenme yeteneğine sahiptir.

$$\mathcal{L}_{D1} = \mathbb{E}_{s,q,b}(\|s - D_1(c_1)\| - \|s - D_1(G(s, b))\|) \quad (2.10)$$

$$\mathcal{L}_{D2} = \mathbb{E}_{c_1,c_2,s,b}(\log D_2(c_1, c_2) + \log(1 - D_2(c_1, G(s, b)))) \quad (2.11)$$



Şekil 2.7 Çok ayırıcılı çekişmeli üretici ağ mimarisine genel bakış [20].

$$\mathcal{L}_G = \mathbb{E}_{s,b,c_1} (\|s - D_1(G(s,b))\| + \log D_2(c_1, G(s,b))) \quad (2.12)$$

Çok ayırıcılı çekişmeli üretici ağ mimarisinin denklemini açıklamadan önce kullanılan semboller açıklamanın anlaşılabilirliği artıracaklarını düşünüyoruz. Kullanılan semboller orijinal makaledeki örnek çalışmadan alınmıştır. s , bir karaktere ait iskelet sistemini temsil etmektedir. b , bir karakterin bedenini yani temelini ifade eder. c_1 ve c_2 ifadeleri karakter veri setinden elde edilen 2 farklı küçük yığından (minibatch) örnekleri temsil etmektedir. Mimari içerisinde iki farklı ayırıcı ağ bulunur. İlk ayırıcı ağ (D_1) görüntüden iskelet sistemi elde etmeyi öğrenir. Öğrenme işlemini gerçekleştirirken Denklem 2.10'da gösterilen maliyet fonksiyonunu kullanmaktadır. Fonksiyonda $s - D_1(c_1)$ ifadesi iskelet örnekleri (s) ile c_1 örneklerinin farkını gösterir. $s - D_1(G(s,b))$ ise üretici ağ tarafından üretilen karakterlerin iskelet örnekleri ile oluşan farkıdır. Ayırıcı ağ iskelet verilerini hem gerçek verilerle hem de sahte olarak üretilen verilerle karşılaştırır. Elde ettiği iki sonucun farkı ile eğitimi sağlamaya çalışır.

Mimarideki ikinci ayırıcı ağ (D_2) renk kullanımını kontrol etmektedir. Renklerin aynı olup olmadığı konusunda ayırım yapmaktadır. Denklem 2.11'de bulunan $D_2(c_1, c_2)$ ifadesinde ayırıcı ağ tarafından c_1 örneği ile c_2 örneğinin renklerinin kıyaslaması yapılır. $D_2(c_1, G(s,b))$ ifadesinde ise c_1 örneği ile üretici ağ tarafından üretilen görüntü karşılaştırılır. Bu sayede üretilen görüntünün rengi değerlendirilir.

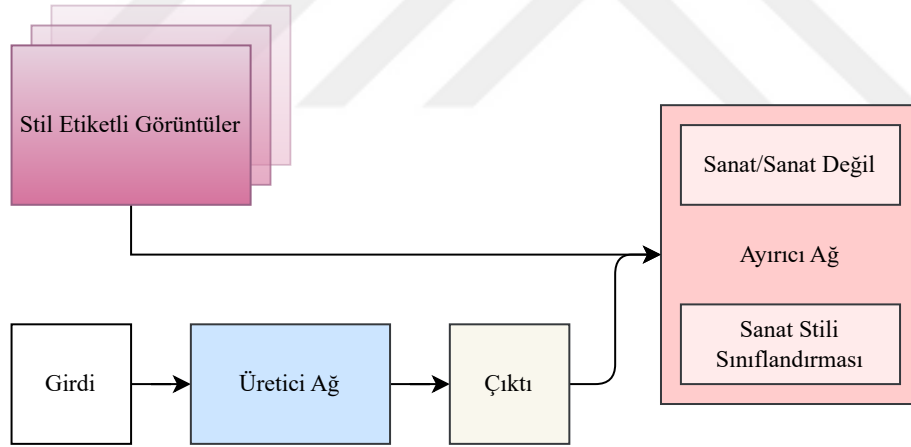
Denklem 2.12'de gösterildiği üzere üretici ağ maliyet fonksiyonu D_1 ve D_2 ayırıcı ağlarında bulunan üretici ağ (G) fonksiyonlarının geçtiği ifadelerin toplamıdır. Bu sayede üretici ağ iki alanda birden gelişim sağlayabilmektedir. Mimaride optimizasyonu sağlamak için ise stokastik gradyan iniş yöntemi kullanılmıştır.

Geliştirilen çok ayırıcılı çekişmeli üretici ağ mimarisi ile animasyon üretimi gerçekleştirmeye çalışılmıştır. Yapılan çalışmada üretici ağ için 2 farklı girdi bulunmaktadır. Bu girişlerde yapılan işlemlerde bilgi kaybının azaltılması için U-Net mimarisi üretici ağa entegre edilmiştir. Üretici ağın birden çok girişi birbiri

ile eşlemeyi öğrenmesi sağlanır. Mimaride bulunan 2 farklı ayırıcı ağ, üretici ağ tarafından üretilen veriyi farklı alan bazında kontrol eder. Bu sayede üretilen görüntü 1 yerine 2 farklı açıdan ilerleme sağlar. Çalışma sonuçlarıyla başarısını ortaya koymuştur.

2.2.7 Yarattıcı Çekişmeli Üretici Ağ (YÇÜA)

Yarattıcı çekişmeli üretici ağ (creative generative adversarial network (CAN)) mimarisi sanatsal çalışmalar ortaya koyabilmesi amacıyla sunulmuş bir çekişmeli üretici ağ mimarisidir (Şekil 2.8) [61]. Yarattıcı çekişmeli üretici ağ mimarisinde girdi görüntünün yanı sıra sanatsal çalışmalar stil etiketleriyle (style label) beraber ayırıcı ağa verilir. Ayırıcı ağ ise girdi olarak verilen görüntünün sanat olup olmadığını kontrol eder. Ayrıca görüntünün verilen stil etiketine uygun bir eser olup olmadığını kontrol eder. Böylelikle ayırıcı ağ girdiyi sınıflandırır ve bunlarla beraber geri dönüş sağlar. Yapılan sınıflandırma ve kontrole göre ilerleme sağlanır. Yarattıcı çekişmeli üretici ağ mimarisi stil etiketlerini üretimde kullanmaz. Stil etiketlerini sınıflandırma yaparken kullanır. Doğal olarak yarattıcı çekişmeli üretici ağ mimarisinde üretim koşullu olarak değerlendirilmemelidir.



Şekil 2.8 Yarattıcı çekişmeli üretici ağ mimarisine genel bakış [61]. Yarattıcı çekişmeli üretici ağ mimarisinde özellikle ayırıcı ağın işlevi çok önemlidir. Ayırıcı ağ, kendisine verilen üretici ağ çıktısının sanat olup olmadığını değerlendirir. Aynı zamanda üretilen görüntünün hangi sanat stiline (art style) ait olduğunu sınıflandırır.

Yarattıcı çekişmeli üretici ağ mimarisi, çekişmeli üretici ağ mimarisine göre daha karmaşık bir yapı ve formüle sahiptir. Elgammal vd. [61] tarafından ortaya konan yarattıcı çekişmeli üretici ağ mimarisinin etkinliğinin ölçülebilmesi amacıyla sanatçıya ait sanatlar ve yarattıcı çekişmeli üretici ağ mimarisi, çekişmeli üretici ağ mimarisi ile üretilen sanatların insanlar üzerinde bıraktığı etki göz önüne alınmıştır. Sonuçlar, başarılı olduğunu göstermektedir.

$$\begin{aligned}
\min_G \max_D V(G, D) = & \mathbb{E}_{x, \hat{c} \sim p_{data}} [\log D_r(x) + \log D_c(c = \hat{c}|x)] + \\
& \mathbb{E}_{z \sim p_z} [\log(1 - D_r(G(z))) - \\
& \sum_{k=1}^K (\frac{1}{K} \log(D_c(c_k|G(z))) + \\
& (1 - \frac{1}{K}) \log(1 - D_c(c_k|G(z))))]
\end{aligned} \tag{2.13}$$

Yaratıcı çekişmeli üretici ağ mimarisinin genel maliyet fonksiyonu Denklem 2.13'te gösterilmiştir. D_r ayırıcı ağın gerçek ve sahte ayrımı yaptığını, D_c ise ayırıcı ağın sınıf etiketlerine uygunluğunu kontrol ettiğini göstermektedir. Dolayısıyla $D_r(x)$ ifadesi gerçek verilerin (x) gerçekliğini kontrol eder. $D_c(c = \hat{c}|x)$ ifadesi ise ayırıcı ağın, gerçek verilerin doğru sınıf etiketlerine sahip olmasını kontrol ettiğini gösterir. Bu sayede ayırıcı ağ, gerçek veri ve bu gerçek veriye ait doğru sınıf etiketlerini tanıır. Sınıf etiketleri ile gerçek veri arasındaki ilişkiyi öğrenir.

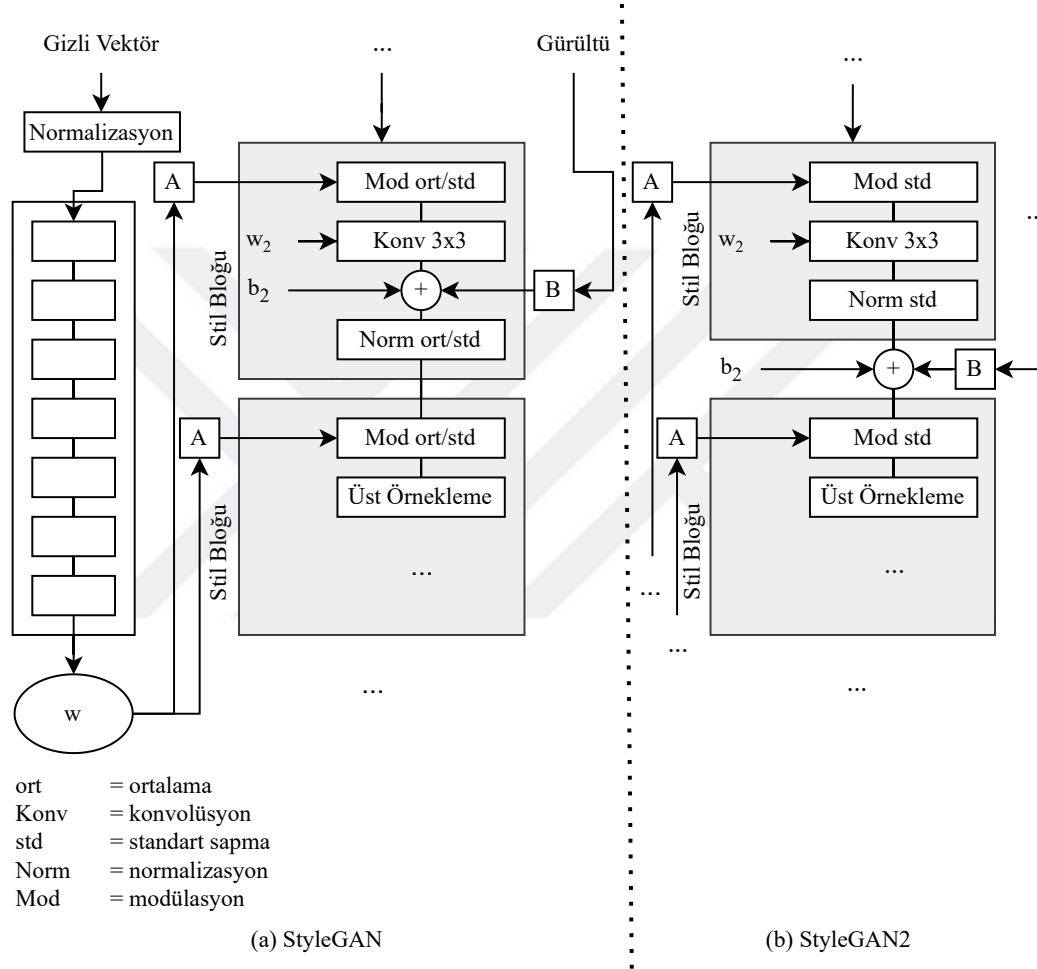
Denklemden $D_r(G(z))$ ifadesi sayesinde ayırıcı ağ, üretici ağın ürettiği veriyi gerçek veya sahte olarak sınıflandırır. z ifadesi mimaride girdi olarak verilen gürültüyü temsil etmektedir. $\frac{1}{K} \log(D_c(c_k|G(z)))$ ifadesi ile üretilen verinin bir sınıfa ait olma olasılığı kontrol edilir. $(1 - \frac{1}{K}) \log(1 - D_c(c_k|G(z)))$ ifadesi ise üretilen verinin sınıfa ait olmama olasılığı kontrol edilir. Mimari, üretilen verinin sınıflara ait olmama olasılığını kontrol ederek, verinin doğru sınıfa ait olma olasılığını arttırmayı amaçlamıştır. Son olarak, mimaride optimizasyon yöntemi olarak Adam yöntemi kullanılmıştır.

Gelecek çalışmalarda bu mimarinin karakter üretiminde kullanılabileceğini öngörüyoruz. 2 boyutlu karakterlerin kullanıldığı alanlarda da tasarımcıların kendine has bir stili bulunur. Üretilebilecek karakterlerin belirli bir stile bağlı kalınarak üretilmesi görüntü olarak bütünlük sağlayacaktır. Yaratıcı çekişmeli üretici ağ mimarisi bu neden göze alındığında karakter üretim süreci otomasyonu için öne çıkabilecek bir yapıya sahiptir.

2.2.8 Stil Çekişmeli Üretici Ağ 2 (StyleGAN2)

Stil çekişmeli üretici ağ 2 (style generative adversarial network 2 (StyleGAN2)) mimarisi stil çekişmeli üretici ağ (style generative adversarial network (StyleGAN)) mimarisinin geliştirilmiş halidir [62, 63]. StyleGAN2 mimarisini açıklamak için kabaca StyleGAN mimarisini de açıklamanın uygun olacağını düşünüyoruz. StyleGAN mimarisinde üretici ağ içerisinde üretim yapılırken klasik bir şekilde gürültü kullanılır ve stil blokları ile işlemler toparlanır (Şekil 2.14.a). Ancak

bu gürültü girişinin yanı sıra gizli uzayı (latent space) da girdi olarak kullanılır. Uzayın içinde bulunan vektörler istenen stile uygun üretim yapılmasını sağlamak için kullanılır. Aynı zamanda uzayın içerisinde bulunan vektörler afin dönüşüm (affine transformation) yardımıyla her konvolüsyon katmanının sonunda kullanılan uyarlanabilir örnek normalizasyon (adaptive instance normalization (AdaIN)) işleminde kullanılır [64]. Son olarak StyleGAN mimarisinde gizli uzay içerisinde bulunan iki farklı vektör karıştırılarak da kullanılabilir.



Şekil 2.9 (a) StyleGAN mimarisindeki üretici ağ [62]. Üretici ağ içerisinde normalizasyon işlemi modülasyondan hemen sonra gerçekleşir. (b) StyleGAN2 mimarisindeki üretici ağ [62]. Yapılan iyileştirme ile beraber normalizasyon işlemi ile modülasyon işleminin bağlantısı koparılır.

$$\text{AdaIN}(x_i, y) = y_{s,i} \frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i} \quad (2.14)$$

Uyarlanabilir örnek normalizasyon yöntemi sayesinde stil transfer edilirken içeriği korunur. Uyarlanabilir örnek normalizasyon yöntemi Denklem 2.14'te gösterilmiştir. x_i girdi verisinin özellik haritasındaki i . kanalı (channel), y ise stili

temsil eder. x_i matrisinden x_i matrisinin ortalaması çıkarılır ve yine x_i matrisinin standart sapmasına bölünür. Böylece normalizasyon sağlanır. $y_{s,i}$ hedef stilin i . kanalına ait matrisin standart sapmasını, $y_{b,i}$ ise hedef stilin i . kanalına ait ortalamasını ifade eder. $y_{s,i}$ ile elde edilen normalizasyon çarpılır ve ölçeklendirilir. Son olarak sonuç $y_{b,i}$ ile toplanır ve kaydırma sağlanır. Bu sayede hedef stile göre girdi verisi yönlendirilecektir.

StyleGAN2 mimarisi ise StyleGAN mimarisinde yapılan değişiklikler ve iyileştirmelerle ortaya çıkmıştır (Şekil 2.9.b) [65]. StyleGAN2 mimarisinde uyarlanabilir örnek normalizasyonu yöntemi 2'ye ayrılarak kullanılmıştır: Normalizasyon (normalization) ve modülasyon (modulation). Bunun yapılmasının nedeni stil bloklarının (style blocks) daha iyi bir şekilde ayrılmasını sağlamaktır. Gürültü ekleme kısmı uyarlanabilir örnek normalizasyon yönteminde bulunan normalizasyon kısmından sonraya alınmıştır ve stil dışında gerçekleşir. Böylece stillerin daha çok vurgulanması sağlanmaya çalışılmıştır. Bunun yanı sıra stil bloklarında kullanılan stil vektörlerinin ağırlıkları ayarlanabilir hale getirilmiştir.

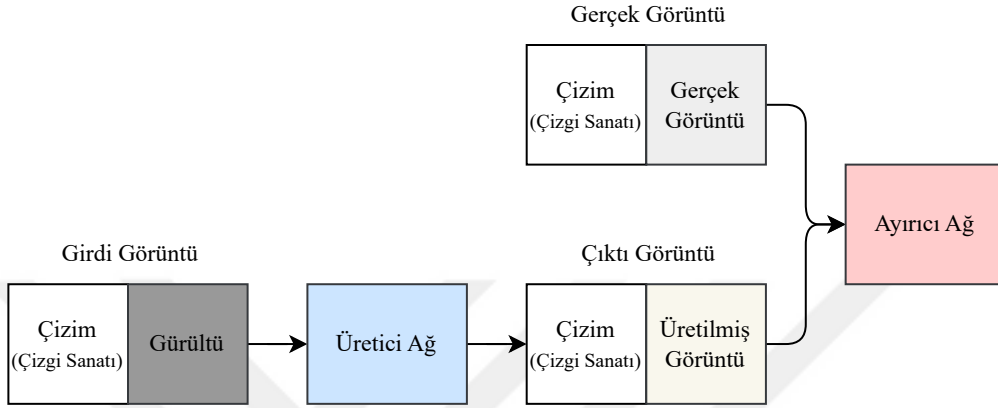
Genel olarak üretici ağ fonksiyonu içerisinde değişiklik yapıldığından, StyleGAN mimarisi geleneksel çekişmeli üretici ağ mimarisi ile aynı genel maliyet fonksiyonuna sahiptir. Ancak üretici ağ (G) fonksiyonunun içeriği değiştirilmiştir. StyleGAN2 mimarisi içinde aynı durum geçerlidir. Bahsettiğimiz değişiklikler sayesinde StyleGAN2 mimarisi veri çeşitlendirmesi için kullanılabilir.

Abdal vd. [66] tarafından yapılan çalışmadan etkilenerek bu mimarilerin veya bu mimarilerden esinlenerek ortaya çıkarılabilecek mimarilerin karakterlere ait varyasyon üretiminde kullanılabileceğini düşünüyoruz. Örneğin hazırlanmış bir karaktere gözlük takılması veya Coutinho ve Chaimowicz tarafından yapılan çalışmada olduğu gibi karakterin farklı pozisyonlara bakan varyasyonlarının elde edilmesi gibi çalışmalar bahsettiğimiz StyleGAN mimarisi ve StyleGAN2 mimarisi ile yapılabilir [55].

2.2.9 Bağlamsal Çekişmeli Üretici Ağ (BÇÜA)

Bağlamsal çekişmeli üretici ağ (contextual generative adversarial network) mimarisi çizim üzerine yoğunlaşan bir mimaridir [67]. Normalde çizim ile uğraşan mimariler, üretici ağ ile üretim yaparken el çizimi veya çizgi sanatından boyanmış görüntüyü elde ederler. Bu görüntü kıyaslanırken, görüntünün gerçek boyaması ile kıyaslanır. Ancak bağlamsal çekişmeli üretici ağ mimarisinde bir bağlam oluşturulmaya çalışılır. Üretici ağ ile üretim yapılırken üretimin yanında verilen çizimde görüntüde bulunur (Şekil 2.10). Kıyaslama yapılırken ise görüntünün

orijinal hali ile çizim halinin birleşiminden oluşan bir görüntü ile kıyaslama yapılır. Aynı zamanda mimaride verilen girdi görüntüsü için üretici ağ (G) ve ayırıcı ağ (D) ağırlıkları değiştirilmeden güncellemeler yapılmaktadır. Yapılan bu değişikliklerle beraber mimarinin el çizimi veya çizgi sanatı konusunda öne çıkabileceğini düşünüyoruz. Bu nedenle gelecek çalışmalarda özellikle karakter renklendirme alanında kullanılabileceğini öngörüyoruz.



Şekil 2.10 Bağlamsal çekişmeli üretici ağ mimarisine genel bakış. Bağlamsal çekişmeli üretici ağ mimarisinde girdi olarak kullanılan görüntüde el çizimi ile üretilecek kısım birleştirilerek verilir.

Bağlamsal çekişmeli üretici ağ mimarisi, üretici ağ ve ayırıcı ağ yapısı olarak genel bir yapı kullanır. Toplu normalleştirme ve konvolüsyon katmanları kullanılır. Üretici ağın son katmanı haricinde aktivasyon fonksiyonu olarak Leaky ReLU kullanılır. Çıkış katmanı için ise tanh aktivasyon fonksiyonu tercih edilmiştir. Ayırıcı ağda ise son katmanda softmax aktivasyon fonksiyonu kullanılmıştır.

$$\mathcal{L}_{contextual}(z) = D_{KL}(\mathbf{M} \odot y, \mathbf{M} \odot G(z)) \quad (2.15)$$

Bağlamsal çekişmeli üretici ağ mimarisi bağlamsal (contextual) ve algısal (perceptual) kayıptan oluşmaktadır. Bağlamsal kayıp Denklem 2.15'te gösterilmiştir. $\mathbf{M}$ maske matrisini temsil etmektedir. Maske matrisini oluştururken, önemli olan görsel noktalarını dikkate almak büyük önem taşımaktadır. $\odot$ sembolü Hadamard çarpımını temsil etmektedir. y girdi görüntüdeki gürültü olmayan kısmı, yani el çizimi veya çizgi sanatı kısmını ifade eder. z ise girdi görüntüdeki gürültüdür. $D_{KL}(\mathbf{M} \odot y, \mathbf{M} \odot G(z))$ ifadesi ayırıcı ağın, üretici ağ tarafından üretilen görüntü ile y verisinin maske ile Hadamard çarpımına tabi tutulması sonucu KL sapmasına (Kullback-Leibler divergence) göre benzerliğini yorumlar [68]. KL sapması, bir olasılık dağılımının ikinci bir olasılık dağılımından nasıl farklılaştığının bir ölçüsüdür [68].

$$D_{KL}(q \parallel p) = \mathbb{E}_{x \sim q(x)} \left[\log \frac{q(x)}{p(x)} \right] \quad (2.16)$$

KL sapması Denklem 2.16 üzerinde gösterilen biçimde hesaplanır. $q(x)$ ve $p(x)$ ifadeleri olasılık dağılımlarını temsil eder. KL sapması, p için optimize edilmiş bir kod kullanarak q 'dan örnekleri kodlamak için gereken beklenen ekstra bit sayısını ölçer. Bu, q 'ya yaklaşmak için p kullanıldığında kaybedilen bilginin bir ölçüsüdür. KL sapması negatif değildir ve yalnızca her yerde $q = p$ ise sıfırdır.

$$\mathcal{L}_{perceptual}(z) = \log(1 - D(G(z))) \quad (2.17)$$

Algısal kayıp Denklem 2.17'de gösterilmiştir. Algısal kayıp ile ayırıcı ağ, üretici ağın ürettiği verinin sahteliğini yorumlar. Bu sayede üretici ağın daha gerçekçi üretimler yapması teşvik edilir.

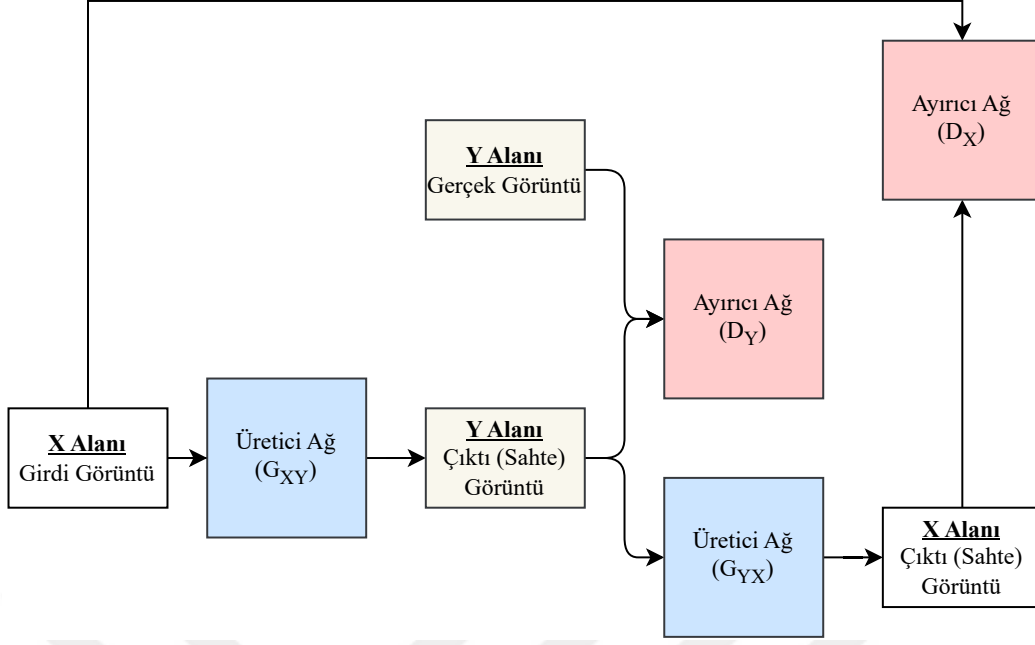
$$\mathcal{L}_{ContextualGAN}(z) = \mathcal{L}_{contextual}(z) + \lambda \mathcal{L}_{perceptual}(z) \quad (2.18)$$

Algısal kayıp ile bağlamsal kayıp toplamı bağlamsal çekişmeli üretici ağ mimarisinin genel maliyet fonksiyonunu oluşturur (Denklem 2.18). λ ile üretilecek görüntünün girdi görüntüye benzerliği ayarlanır. λ değerinin düşük olması daha benzer görüntüler oluşmasını sağlayacaktır. Bağlamsal çekişmeli üretici ağ mimarisinde optimizasyon yöntemi olarak Adam yöntemi tercih edilmiştir.

2.2.10 Döngü Tutarlı Çekişmeli Üretici Ağ (CycleGAN)

Döngü tutarlı çekişmeli üretici ağ (cycle-consistent generative adversarial network (CycleGAN)) mimarisi Pix2Pix mimarisini baz alarak ortaya çıkmış bir mimaridir [69]. CycleGAN mimarisi sayesinde giriş görüntüsünden beklenen çıkış elde edilebilirken, aynı zamanda çıktı görüntüden giriş görüntüsünün tarzında bir görüntü de elde edilebilir (Şekil 2.11). Dolayısıyla çapraz öğrenme sağlanmış olur. Mimarinin temel amacı ise görüntünün eşlenmeden uygulanmasını sağlamaktır. Pix2Pix mimarisinde her bir görüntü için gerçek görüntünün de eklenmesi gereklidir. Girdi görüntü ile elde edilmesi beklenen görüntü eşlenmiş olur. Ancak CycleGAN mimarisinde 2 farklı veri seti eşlenmeden kullanılabilir.

CycleGAN mimarisinde 2 üretici ağ ve 2 ayırıcı ağ bulunmaktadır. Bu sayede çapraz öğrenme sağlanır. Mimaride bulunan üretici ağda artık sinir ağı blokları kullanılır [60]. Tüm mimari boyunca Pix2Pix mimarisinde tercih



Şekil 2.11 CycleGAN mimarisine genel bakış [69].

edilen toplu normalleştirme yerine örnek normalizasyon kullanılmaktadır [70]. Kayıp fonksiyonu olarak döngü tutarlılık kaybı (cycle consistency loss) kullanılır. Oluşturulan bu mimari ile beraber görüntüler eşlenmeden aralarında dönüşüm yapılabilir.

$$\mathcal{L}_{GAN}(G_{XY}, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)} [\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)} [\log(1 - D_Y(G_{XY}(x)))] \quad (2.19)$$

$$\mathcal{L}_{GAN}(G_{YX}, D_X, Y, X) = \mathbb{E}_{x \sim p_{data}(x)} [\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)} [\log(1 - D_X(G_{YX}(y)))] \quad (2.20)$$

CycleGAN mimarisinde alanlar arası üretim yaparken kullanılan çekişme kaybı (adversarial loss) fonksiyonu örneği Denklem 2.19'da gösterilmiştir. Denklem X alanından Y alanına üretim yaparken kullanılır ve Denklem 2.5'te tanıttığımız çekişmeli üretici ağ denklemine oldukça benzemektedir. Y alanında üretim yapılmak istendiği için ayırıcı ağ, Y alanındaki verilere göre ayırım yapmaktadır. Y alanındaki veriden X alanındaki bir veri elde etmek istendiğinde kullanılan çekişme kaybı ise Denklem 2.20'de gösterilmiştir.

CycleGAN mimarisi X ve Y alanlarında veri üretmeyi sağlayan çekişme kaybı fonksiyonu dışında döngü tutarlılık kaybı içerir. Döngü tutarlılık kaybı sayesinde bir alandan diğer alana çevrilen görüntü, tekrar orijinal alanına çevrildiğinde orijinal

görüntüye yakınlık sağlanmaya çalışılır.

$$\begin{aligned}\mathcal{L}_{cyc}(G_{XY}, G_{YX}) = & \mathbb{E}_{x \sim p_{data}(x)} [\|G_{YX}(G_{XY}(x)) - x\|_1] + \\ & \mathbb{E}_{y \sim p_{data}(y)} [\|G_{XY}(G_{YX}(y)) - y\|_1]\end{aligned}\quad (2.21)$$

Döngü tutarlılık kaybı Denklem 2.21’de belirtilmiştir. $G_{YX}(G_{XY}(x)) - x$ ifadesi ile X alanındaki bir görüntüden önce $G_{XY}(x)$ ile Y alanında bir görüntü elde edilir. $G_{YX}(y)$ olarak elde edilen ifade ile Y alanından tekrar X alanında görüntü üretilir. Son olarak orijinal alanına döndürülen görüntü ile orijinal görüntü (x) arasındaki fark L1 norm yöntemiyle incelenir. Yapılan inceleme ile işlenerek orijinal alanına döndürülen görüntünün orijinal görüntü ile benzer kalması sağlanmaya çalışılır.

$$\begin{aligned}\mathcal{L}(G_{XY}, G_{YX}, D_X, D_Y) = & \mathcal{L}_{GAN}(G_{XY}, D_Y, X, Y) + \\ & \mathcal{L}_{GAN}(G_{YX}, D_X, Y, X) + \\ & \lambda \mathcal{L}_{cyc}(G_{XY}, G_{YX})\end{aligned}\quad (2.22)$$

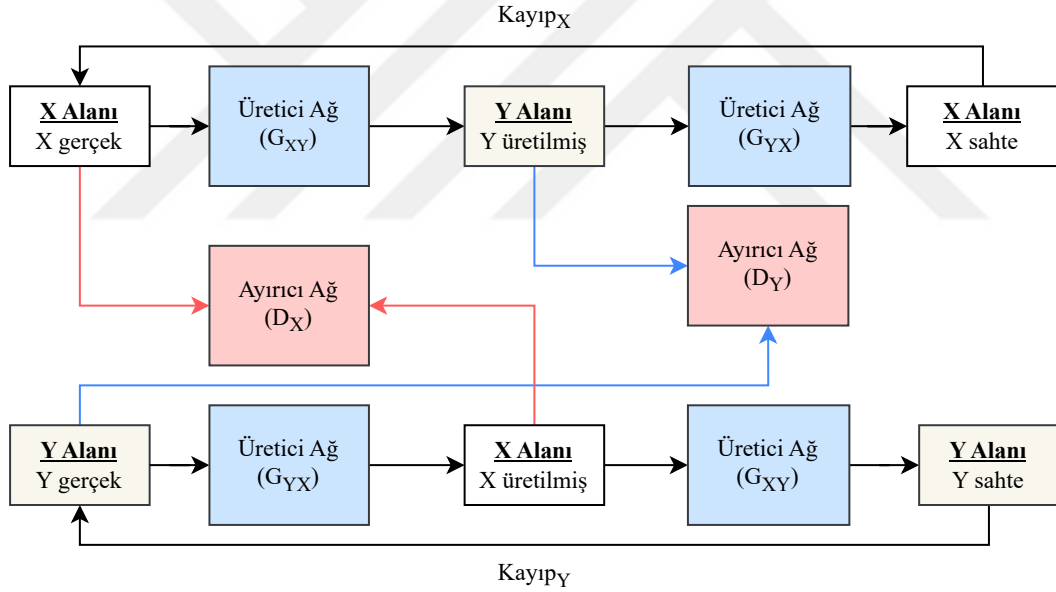
CycleGAN mimarisinin genel maliyet fonksiyonu ise bahsettiğimiz tüm kayıp değerlerinin toplamıdır (Denklem 2.22). λ parametresi mimarideki tutarlılığı kontrol eder. λ parametresi küçük ise döngü tutarlılık kaybının önemi azalır. Bu nedenle üretilen görüntüler daha gerçekçi olurken dönüşümlerde zayıflık yaşanması beklenir. λ parametresinin değeri büyük olduğunda ise dönüşümlerdeki tutarlılığın yüksek olması ancak üretilen görüntülerin gerçekçiliğinde problemler olması beklenir. CycleGAN mimarisi optimizasyon yöntemi olarak ise Adam yöntemini kullanır.

Wong [71] CycleGAN mimarisi ile karakterlere ait varyasyon üretme çalışması yapmıştır. Yaptığı bu çalışmada Pokémon karakterlerine ait veri setini kullanmıştır. Pokémon dünyasında karakterler sahip oldukları element tipine göre farklı renk paletinde bir çizime sahiptirler. Yapılan bu çalışma ile Wong Pokémon karakterlerinin element tiplerine göre dönüşümlerini elde etmeye çalışmış ve başarmıştır. Bu çalışma CycleGAN mimarisinin varyasyon için kullanılabileceğini kanıtlar niteliktedir.

2.2.11 Alanlar Arası İlişkileri Keşfeden Çekişmeli Üretici Ağ (DiscoGAN)

Alanlar arası ilişkileri keşfeden çekişmeli üretici ağ (Discover Cross-Domain Relations with Generative Adversarial Networks (DiscoGAN)), iki farklı alan arasında çapraz öğrenmeyi sağlayan iki üretici ağı ve iki ayırıcı ağı olan bir

mimaridir (Şekil 2.12) [72]. Üretici ağ girdiden yeni veriler üretir ve iki farklı üretici ağ vardır [73]. X ve Y olarak gösterilen iki alanı ele alarak örnek üzerinden DiscoGAN mimarisini açıklamak anlaşılabilirliği arttıracaktır. X alanında bulunan ve X gerçek olarak adlandırılan veriler, bir klasik çekişmeli üretici ağ mimarisi kullanılarak Y alanına dönüştürülür ve Y alanında üretilmiş Y üretilmiş verisi elde edilir. Üretilen Y verisi daha sonra tekrar X alanına dönüştürülerek X sahte verisi elde edilir. X gerçek ve X sahte arasındaki karşılaştırma, çekişmeli üretici ağ mimarisini eğitmek için kayıp fonksiyonda kullanılır [74]. Bu süreç üretici ağlardan birini oluşturur. Diğer üretici ağ da benzer şekilde çalışır, Y alanından veri alır (Y gerçek olarak adlandırılır), X üretilmiş adı verilen X alanında bir veri üretir ve ardından Y sahte verisi üretir. Kayıp fonksiyonu, Y gerçek ve Y sahte karşılaştırılarak hesaplanır. Özetle, DiscoGAN mimarisinin her biri aynı süreci izleyen iki üretici ağı vardır. DiscoGAN üretici ağları klasik çekişmeli üretici ağ mimarisini kullandığı için klasik çekişmeli üretici ağ mimarisini anlamak DiscoGAN mimarisini anlamak için çok önemlidir.



Şekil 2.12 DiscoGAN mimarisine genel bakış [72].

DiscoGAN iki üretici ağ içerdiği gibi iki ayırtıcı ağ içerir. Ayırtıcı ağlardan biri, kayıp fonksiyonu ve eğitimi için girdi olarak Y üretilmiş ve Y gerçek olarak adlandırdığımız verilerini kullanır. Diğer ayırtıcı ağ, eğitimi için X üretilmiş ve X gerçek olarak adlandırdığımız verileri kullanır.

$$\mathcal{L}_{GAN_Y} = -\mathbb{E}_{x \sim p_x} [\log D_Y(G_{XY}(x))] \quad (2.23)$$

$$\mathcal{L}_{CONST_X} = d(G_{YX} \circ G_{XY}(x), x) \quad (2.24)$$

DiscoGAN mimarisi, üretici ağ için klasik çekişmeli üretici ağ kaybını kullanır (Denklem 2.23). CycleGAN mimarisine benzer olmakla birlikte döngü tutarlılık kaybı içermez. Döngü tutarlılık kaybı yerine yeniden üretim kaybı (reconstruction loss) olarak isimlendirilen kayıp fonksiyonunu içerir (Denklem 2.24). Yeniden üretim kaybı X alanından Y alanına dönüştürülen verinin tekrar X alanına döndürülmesi sonucu, orijinal X alanındaki görüntü ile Y alanından dönüştürülerek X alanında elde edilen görüntünün farkını ele alır.

$$\mathcal{L}_{GAN_X} = -\mathbb{E}_{y \sim p_y} [\log D_X(G_{YX}(y))] \quad (2.25)$$

$$\mathcal{L}_{CONST_Y} = d(G_{XY} \circ G_{YX}(y), y) \quad (2.26)$$

$$\mathcal{L}_G = \mathcal{L}_{GAN_Y} + \mathcal{L}_{CONST_X} + \mathcal{L}_{GAN_X} + \mathcal{L}_{CONST_Y} \quad (2.27)$$

İki farklı alan için dönüşümler yapmaya odaklı bir mimari olan DiscoGAN mimarisinde iki üretici ağ vardır. Diğer üretici ağ ise Denklem 2.25'te belirtilen kayıp fonksiyonunu ve Denklem 2.26'da gösterilen yeniden üretim kaybını kullanır. Mimarideki toplam üretici ağ kayıp fonksiyonu Denklem 2.27'de gösterildiği gibi tüm üretici ağ denklemlerinin toplamını içerir.

$$\mathcal{L}_{D_X} = -\mathbb{E}_{x \sim P_x} [\log D_X(x)] - \mathbb{E}_{y \sim P_y} [\log(1 - D_X(G_{YX}(y)))] \quad (2.28)$$

$$\mathcal{L}_{D_Y} = -\mathbb{E}_{y \sim P_y} [\log D_Y(y)] - \mathbb{E}_{x \sim P_x} [\log(1 - D_Y(G_{XY}(x)))] \quad (2.29)$$

$$\mathcal{L}_D = \mathcal{L}_{D_X} + \mathcal{L}_{D_Y} \quad (2.30)$$

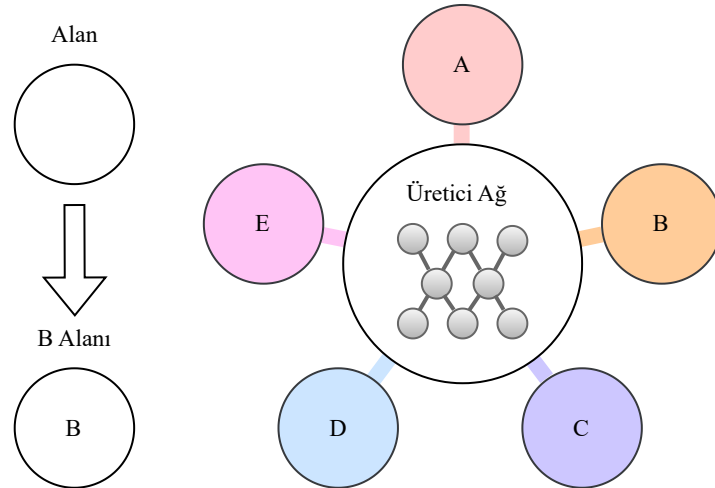
DiscoGAN mimarisi, ayırıcı ağ kaybı olarak klasik çekişmeli üretici ağ mimarisindeki ayırıcı ağ kaybını kullanmaktadır. X alanında kullanılan ayırıcı ağın kayıp fonksiyonu Denklem 2.28'de gösterilmiştir. Mimaride iki ayırıcı ağ bulunduğu için Y alanı için kullanılan diğer ayırıcı ağa ait kayıp fonksiyonu ise

Denklem 2.29’da sunulmuştur. İki ayırıcı ağa ait kayıp fonksiyonlarının toplamı mimariye ait ayırıcı ağ kaybını ifade eder (Denklem 2.30). DiscoGAN mimarisinde optimizasyonu sağlamak için Adam yöntemi kullanılmıştır.

DiscoGAN mimarisinin karakter üretim aşamasında yapılabilecek olan çalışmalarda kullanabileceğini öngörüyoruz. Çünkü yaptığı çapraz öğrenme işleminin bir sağlama gibi olduğu düşünüyoruz. Dolayısıyla çizgi sanatı görüntüsünden boyalı bir görüntü elde edildiğinde, bu görüntünün tekrar çizgi sanatı görüntüye çevrilmesi ile sağlama yapılmış olacaktır. DiscoGAN mimarisinin sağladığı bu özellik, karakter tasarımcılarının çizgi sanatı görüntüsünü manipüle etmelerine ve ayarlamalar yapmalarına olanak tanıyarak onlara daha fazla kontrol ve esneklik sağlar. Özellikle kontrol sağlamasından dolayı çapraz öğrenme özelliğine sahip olması oldukça önemlidir.

2.2.12 Yıldız Çekişmeli Üretici Ağ (StarGAN)

Yıldız çekişmeli üretici ağ (Star generative adversarial network (StarGAN)) mimarisi farklı alanları tek başına öğrenebilen bir çekişmeli üretici ağ mimarisidir [75]. Farklı alanları ve verilen farklı alanların arasındaki etkileşimi öğrenir (Şekil 2.13). Böylelikle farklı alanların aynı anda işlenmesi için gereken model karmaşıklığından sıyrılır. StarGAN mimarisinde ayırıcı ağ, hem sınıflandırma hem de gerçek ve sahte veri ayrımını yapar.



Şekil 2.13 StarGAN mimarisinde kullanılan üretici ağ [75]. Üretici ağ içerisinde farklı alanlar arasında bağlantı kurulur.

StarGAN mimarisi sabit bir dönüşüm öğrenmek yerine görüntüyü esnek bir şekilde farklı alanlara çevirmeyi öğrenir. Çevirme işlemini gerçekleştirebilmek için alan bilgileri kullanılır. Alan bilgileri etiketlerle temsil edilir, eğitim sırasında model etiketi kullanarak esnek dönüşümü sağlar.

$$\mathcal{L}_{adv} = \mathbb{E}_x [\log D_{src}(x)] + \mathbb{E}_{x,c} [\log(1 - D_{src}(G(x, c)))] \quad (2.31)$$

StarGAN çekişme kaybı, sınıflama kaybı (classification loss) ve yeniden üretim kaybından oluşan maliyet fonksiyonuna sahiptir. Çekişme kaybı Denklem 2.31’de gösterilmiştir. Çekişme kaybı ile üretilen görüntüler gerçeğe yaklaştırılmaya çalışılır. $D_{src}(x)$ ifadesi ile ayırıcı ağ girdiyi temsil eden x verisini değerlendirir. $D_{src}(G(x, c))$ ifadesi ile c alan bilgisi ve x girdisi ile üretici ağ bir veri üretir. Daha sonra ayırıcı ağ bu veriyi değerlendirir. Çekişmeli üretici ağ mimarilerinde bulunan kayıp fonksiyonlarına oldukça benzer bir kayıp fonksiyonudur.

$$\mathcal{L}_{cls}^r = \mathbb{E}_{x,c'} [-\log D_{cls}(c'|x)] \quad (2.32)$$

$$\mathcal{L}_{cls}^f = \mathbb{E}_{x,c} [-\log D_{cls}(c|G(x, c))] \quad (2.33)$$

Sınıflama kaybı olarak iki farklı kayıp kullanılır. Ayırıcı ağ tarafından Denklem 2.32’de gösterilen ve ayırıcı ağın gerçek girdileri orijinal alan bilgilerine (c') göre sınıflandırmayı öğrenmesini sağlayan sınıflama kaybı kullanılır. Üretici ağ ise Denklem 2.33’te gösterilen sınıflama kaybını kullanır. Üretici ağ tarafından kullanılan sınıflama kaybında üretici ağ tarafından alan bilgisi ve girdi veriden üretilen veriler sınıflandırılır. Üretici ağ denklemi minimize ederek sınıflandırmanın doğruluğunu sağlamaya çalışır.

$$\mathcal{L}_{rec} = \mathbb{E}_{x,c,c'} [\|x - G(G(x, c), c')\|_1] \quad (2.34)$$

StarGAN mimarisinde üretici ağ tarafından kullanılan yeniden oluşturma kaybı Denklem 2.34’de gösterilmiştir. $G(G(x, c), c')$ ifadesi ile öncelikle üretici ağa x girdi verisi ve hedef alan verisi (c) verilir. Hedef alanda üretilen veri, x girdisine ait orijinal alan bilgisi olan c' ile beraber üretici ağa verilerek orijinal alanda tekrardan üretilir ve girdi verisi ile farkı alınarak, görüntünün yeniden oluşturulduğunda ne kadar iyi oluşturulduğu incelenmiş olur. L1 normu kullanılarak fark kontrolü yapılır.

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls} \mathcal{L}_{cls}^r \quad (2.35)$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls} \mathcal{L}_{cls}^f + \lambda_{rec} \mathcal{L}_{rec} \quad (2.36)$$

Tüm denklemleri açıkladığımız StarGAN mimarisinde ayırıcı ağ maliyet fonksiyonu Denklem 2.35’te, üretici ağ maliyet fonksiyonu ise Denklem 2.36’da gösterilmiştir. λ_{cls} ile λ_{rec} birer hiper parametredir (hyperparameter). λ_{cls} sınıflandırmanın ve λ_{rec} ise yeniden oluşturma’nın önemini ayarlamak için kullanılır. Mimarinin açıklandığı orijinal makaledeki tüm deneylerde λ_{cls} parametresi 1 değeri ile, λ_{rec} ifadesi 10 değeri ile kullanılmıştır. StarGAN mimarisinde optimizasyon Adam yöntemi ile sağlanmıştır.

StarGAN mimarisinin karakterlerle ilgili yapılabilecek gelecek çalışmalarda kullanılmaya yatkın olduğunu düşünüyoruz. Çünkü oyun veya çizgi roman alanlarında oluşturulan karakterlerin teması ve çizim tarzı tutarlı olmalıdır. Dolayısıyla büyük bir veri kümesinde, aynı tarz görüntüler sınıflandırılabilir. Renk ve tarz olarak uyumu sağlanabilir. Bu sayede üretilen karakterler birbirinden ayrı dünyalara hitap etmemiş olacaktır. Bunun bütünlüğü sağlamak açısından oldukça önemli olduğunu düşünüyoruz.

2.3 Varyasyonel Otomatik Kodlayıcı (VOK) Mimarileri

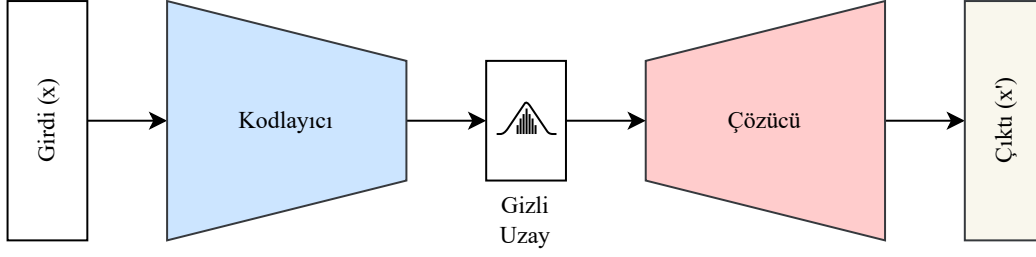
İlk olarak klasik varyasyonel otomatik kodlayıcı mimarisini tanıttık ve ardından problemimizle ilgili varyasyonel otomatik kodlayıcı versiyonlarını gözden geçirdik.

2.3.1 Klasik Varyasyonel Otomatik Kodlayıcı (VOK)

Varyasyonel otomatik kodlayıcı (variational autoencoder (VAE)) mimarisi olasılık dağılımlarını ve klasikleşmiş otomatik kodlayıcı (autoencoder) mimarisinden yararlanarak yeni veriler üretmeyi sağlar (Şekil 2.14) [76–79]. Otomatik kodlayıcı mimarisi içerisinde bir kodlayıcı ve bir çözücü bulundurulur. Kodlayıcı ile veri sıkıştırılır ve gizli uzay vektörü oluşturulur. Çözücü ile gizli uzaydan yararlanılarak veri en az kayıpla tekrar üretilmeye çalışılır. Varyasyonel otomatik kodlayıcı ile sıkıştırılan veriden olasılıksal dağılımlar yardımıyla yeni veriler elde edilebilmektedir.

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL} [q_\phi(z|x) || p_\theta(z)] \quad (2.37)$$

Varyasyonel otomatik kodlayıcı maliyet fonksiyonu Denklem 2.37’de gösterilmiştir. ϕ parametreleri ifade eder. $q_\phi(z|x)$ girdi olan x verisinin z dağılımına



Şekil 2.14 Varyasyonel otomatik kodlayıcı mimarisine genel bakış.

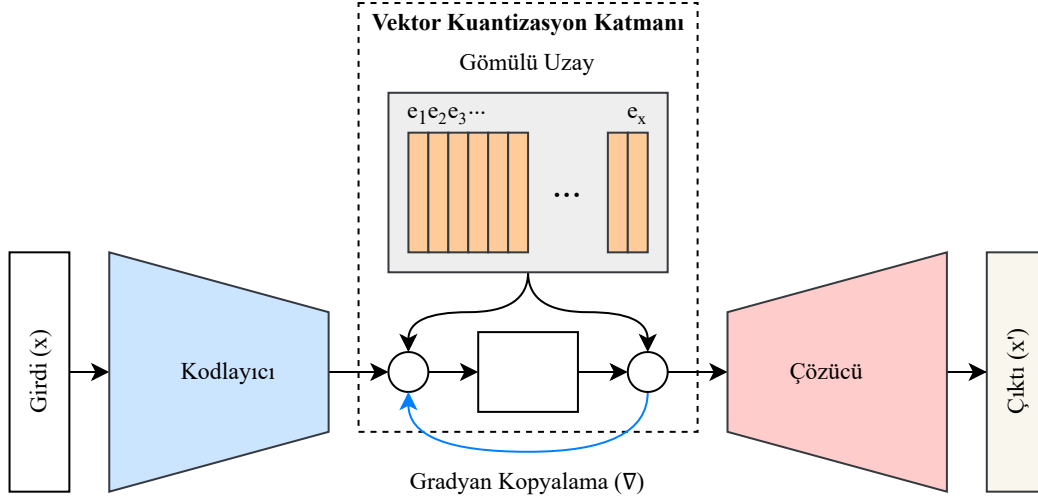
dönüşmesini temsil eder. Dönüştürme yapılırken ϕ parametreleri dikkate alınır. $p_\theta(x|z)$ ifadesi ile z dağılımından x verisi tekrar elde edilir. $D_{KL}[q_\phi(z|x)||p_\theta(z)]$ ifadesi, elde edilen z dağılımının, modelde seçilen z dağılımıyla arasındaki KL sapması farkını ölçer [68]. KL sapması yöntemine Bölüm 2.2.9 değinilmiştir. Varyasyonel otomatik kodlayıcı mimarisi Denklem 2.37'yi kullanarak yeni veriler oluşturur.

2.3.2 Vektör Kuantize Edilmiş Varyasyonel Otomatik Kodlayıcı (VK-VOK)

Vektör kuantize edilmiş varyasyonel otomatik kodlayıcı (vector quantized variational encoder (VQ-VAE)) mimarisi varyasyonel otomatik kodlayıcı mimarisini temel alır [80]. Varyasyonel otomatik kodlayıcı mimarisinde de bulunan kodlayıcı ve çözücü katmanları arasında vektör kuantizasyon katmanı (vector quantization layer) bulundurulur. Vektör kuantizasyon katmanında bir kod defteri (codebook) bulunur. Kod defteri eğitim boyunca güncellenerek veriler için ortalama temsil değerleri içerir. Mimariye verilen girdi bu kod defterindeki en yakın indeks (index) ile kuantize edilir. Bu işlemler sonucunda kuantize edilen veri çözücüde kullanılır ve çözücü girdi verisine benzer yeni bir veri üretir. Vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisinin içerisinde bulunan kod defteri, olasılık dağılımlarına bağımlılıktan kaçınmaya yardımcı olur [81]. Ancak, vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisi olasılıklardan bağımsız hale gelmez. Kod defteri ile birlikte, vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisi olasılıkların yönlendirilmesine izin verir.

$$\mathcal{L}_{VQ-VAE} = \log p(x|z_q(x)) + \|\mathbf{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \mathbf{sg}[e]\|_2^2 \quad (2.38)$$

Vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisi Denklem 2.38'de gösterilen fonksiyon ile üretimi gerçekleştirmektedir. x girdi olarak verilen veriyi



Şekil 2.15 Vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisine genel bakış.

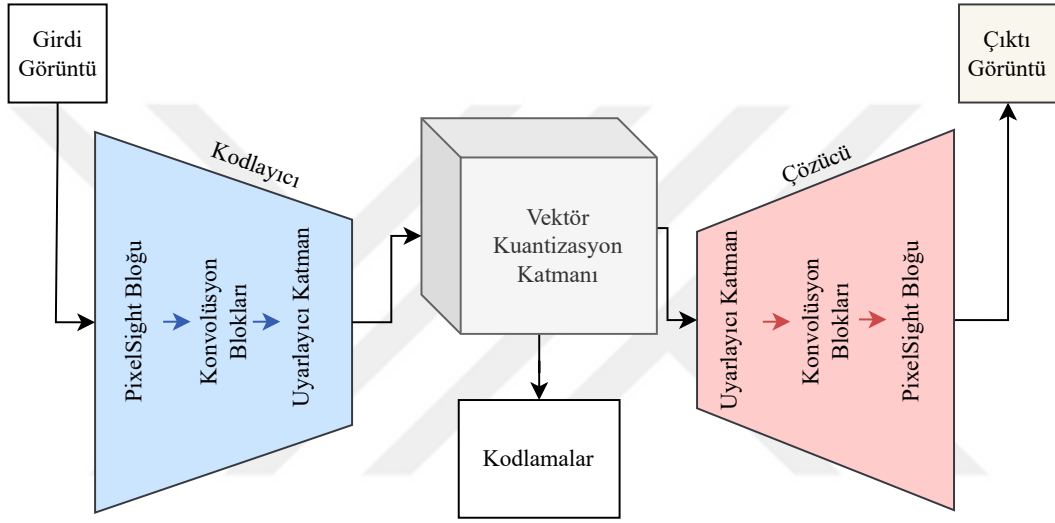
ifade eder. e ise gömülü uzay (embedded space) içerisindeki vektörleri temsil eder. $p(x|z_q(x))$ ifadesi yeniden üretim kaybını ifade eder. Modelin yeniden oluşturduğu veriyi ne kadar iyi seviyede oluşturduğunu inceler. $\|\mathbf{sg}[z_e(x)] - e\|_2^2$ ifadesindeki $\mathbf{sg}$ gradyan durduran fonksiyondur (stop gradient function). Gradyan durduran fonksiyon sayesinde ara değişkenler belirli değerlerde sabit tutulabilmektedir. Dolayısıyla $\mathbf{sg}[z_e(x)]$ ifadesinde de belirtildiği üzere, x giriş verisi için kodlayıcının çıktısını temsil eden $z_e(x)$ değeri gradyan hesaplamalarından etkilenmez. Bu sayede gömülü uzay içerisinde güncelleme sağlanırken, kodlayıcı çıkışları güncellenmeyecektir.

İfadenin tamamında gömülü uzay içerisindeki en yakın vektöre kuantize yapıldığı belirtilir ve minimize edilmeye çalışılır (kod defteri kaybı (codebook loss)). Son olarak $\beta \|z_e(x) - \mathbf{sg}[e]\|_2^2$ ifadesinde gizli uzaya dönüştürülen girdinin gömülü uzay içerisinde bulunan vektöre yakın olması sağlanmaya çalışılır (bağlılık kaybı (commitment loss)). β parametresi bağlılık kaybının önemini kontrol edebileceğimiz bir parametredir ve genel olarak 0.25 değeri ile kullanılır. Vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisinde Adam optimizasyon yöntemi kullanılır [82].

2.3.3 Piksel Vektör Kuantize Edilmiş Varyasyonel Otomatik Kodlayıcı (Pixel VQ-VAE)

Piksel vektör kuantize edilmiş varyasyonel otomatik kodlayıcı (pixel vector quantized variational encoder (Pixel VQ-VAE)) mimarisi, vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisini baz alan bir mimaridir [83]. Saravanan

ve Guzdial [83] tarafından ortaya atılan bu mimaride VK-VOK mimarisine ek olarak 2 ek katman daha bulundurulur (Şekil 2.16). Ek olarak bulunan katmanlardan ilki her pikseli ayrı ayrı değerlendirmek ve model karmaşıklığını azaltmak adına kullanılan PixelSight bloğudur. PixelSight bloklarının içerisinde 1x1 evrişimler (konvolüsyonlar) bulunur. PixelSight bloklarında ReLU aktivasyon fonksiyonu ve toplu normalleştirme kullanılmıştır. Diğer eklenen katman ise uyarlayıcı katmandır (adapter layer). Uyarlayıcı katman VK-VOK mimarisinde bulunan bir problemi çözer. Sıkıştırılmış halde bulunan verinin açılması sonucu verinin boyutu sınırlandırılırken kullanılır. Pixel VQ-VAE mimarisinde optimizasyonu sağlamak amacıyla ise Adam optimizer kullanılır.



Şekil 2.16 Piksel vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisine genel bakış [83].

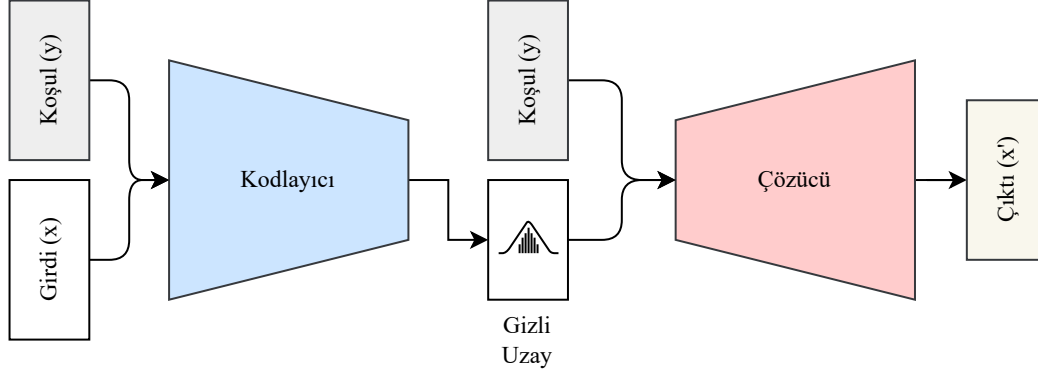
Pixel VQ-VAE mimarisi genel olarak vektör kuantize edilmiş varyasyonel otomatik kodlayıcı mimarisi ile aynı denklemi kullanmaktadır. Mimarinin temel farkı kullandığı ekstra katmanlarla piksel odaklı çalışmasıdır. Dolayısıyla piksel bazındaki değişimlerde daha iyi sonuçlar vermektedir.

Saravanan ve Guzdial [83] ortaya koyduğu bu mimari ile piksel çizimlerinin üzerine yoğunlaşmışlardır. Karakterin farklı varyasyonlarını üretme ve karakter oluşturma üzerine denemeleri bulunmaktadır. Bu denemelerin başarılı sonuçlar verdiğini göstererek, ortaya çıkardıkları mimarinin önemini ortaya koymuşlardır.

2.3.4 Koşullu Varyasyonel Otomatik Kodlayıcı (KVOK)

Koşullu varyasyonel otomatik kodlayıcı (conditional variational autoencoder (CVAE)) mimarisi varyasyonel otomatik kodlayıcı mimarisini temel alarak geliştirilmiştir. Koşullu varyasyonel otomatik kodlayıcı mimarisi bir koşula bağlıdır

ve üretilen verinin yönlendirilmesini olanaklı kılar (Şekil 2.17) [84]. Girdi olarak verilen görüntünün yanı sıra koşulun da mimariye verilmesi gerekir. Aynı zamanda verilen koşul çözücü ile işleme girerken de kullanılır. Bu sayede koşula yönelik bir üretim yapılması beklenir.



Şekil 2.17 Koşullu varyasyonel otomatik kodlayıcı mimarisine genel bakış.

$$\mathcal{L}_{CVAE} = -\mathbb{E}_{q_{\theta}(z|x,y)} [\log p_{\phi}(y|x,z)] + D_{KL} [q_{\theta}(z|x,y) || p_{\phi}(z|x)] \quad (2.39)$$

Koşullu varyasyonel otomatik kodlayıcı mimarisi Denklem 2.39’de gösterilen fonksiyonu kullanmaktadır [85, 86]. x mimariye verilen girdi, y hedeflenen veridir, yani koşul olarak yorumlanabilir. z gizli değişkendir (latent variable). $p_{\phi}(y|x,z)$ ifadesi x ve z girdisi ile oluşturulan y verisinin incelenmesini temsil eder. İnceleme ile y verisinin ne kadar iyi bir çıktı olduğu incelenir. $D_{KL} [q_{\theta}(z|x,y) || p_{\phi}(z|x)]$ üretilen gizli değişken (z) değerinin önceden belirlenen dağılımla olan farkını ölçer. Bu sayede değerler arasındaki yakınlık ölçülür. Amaç, önceden belirlenen dağılıma yaklaşılarak mimarinin performansını iyileştirmektir. Koşullu varyasyonel otomatik kodlayıcı mimarisi optimizasyon yöntemi olarak stokastik gradyan iniş yöntemini kullanır.

Gonzalez vd. [35] koşullu varyasyonel otomatik kodlayıcı mimarisini kullanarak Pokémon oyununa ait karakterlerin farklı varyasyonlarını üretmeye çalışmışlardır. Koşullu varyasyonel otomatik kodlayıcı mimarisinde görüntülere ait tekil kodlama (one-hot-encoded) yöntemiyle çevrilen etiketlerle koşullanma sağlanmıştır. Yapılan bu çalışma ile varyasyon üretme sürecinde bir gelişme sağlanmıştır. Ancak henüz oyun geliştirme süreci için uygun olmadığından bahsedilmektedir.

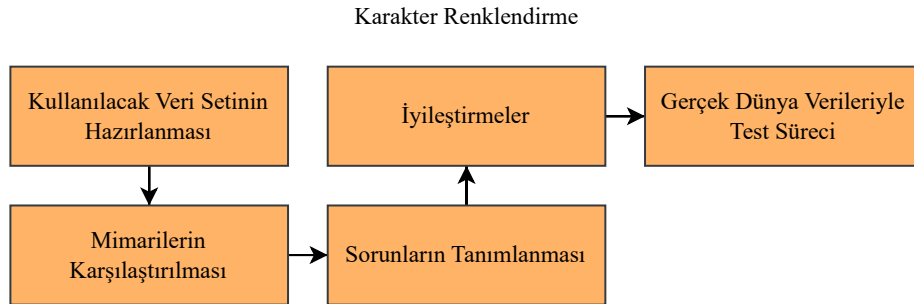
3

DENEYLER

Bu bölümde karakter üretimi sürecinde kullanılan çeşitli mimarileri karşılaştırdık ve yaptığımız iyileştirmeleri açıkladık. Karakter üretim sürecinin iki farklı alanında deneyler gerçekleştirdik: Karakter renklendirme ve karakter üretimi. Bu deneylerden elde ettiğimiz bulguları paylaştık ve sonuçları değerlendirdik.

3.1 Karakter Renklendirme

Karakter renklendirme zorlu bir süreç olmasının yanı sıra oldukça zahmetli bir süreçtir. Bu sürecin otomatikleştirilmesi, bu alanda çalışan veya vakit harcayan insanlar için oldukça önemlidir. Karakter renklendirme alanında yapılan çalışmaları incelerken, özellikle Pix2Pix mimarisine ve onun üzerinde yapılan değişikliklere yoğunlaştık. Pix2Pix mimarisinin ilerlemelerin temelini oluşturduğunu ve karakter renklendirmesi için en ideal yaklaşımlardan biri olduğunu düşünüyoruz.



Şekil 3.1 Çalışmamızın akış diyagramı.

Karakter renklendirme işleminde deneylerimizi yaparken kullandığımız mimariler:

- Coelho vd. [43] tarafından ortaya konan MANGAN mimarisi (Bölüm 2.2.3).
- Isola vd. [45] tarafından ortaya konan orijinal Pix2Pix mimarisi (Bölüm 2.2.4).

- Coutinho ve Chaimowicz [55] tarafından ortaya konan Pix2Pix mimarisi (Bölüm 2.2.4).
- Serpa ve Rodrigues [49] tarafından ortaya konan Pix2Pix mimarisi (Bölüm 2.2.4).
- Jiang ve Sweetser [52] tarafından ortaya konan Pix2Pix mimarisi (Bölüm 2.2.4).

MANGAN mimarisi dışında kalan mimarilerde Pix2Pix mimarisi üzerine yoğunlaştık.

Çalışmalarından yararlandığımız makalelerde ortaya konan mimariler, karakter renklendirmek amacıyla kullanıldı. Mimarileri kullanırken mimarilerin her biri için en iyi sonucu elde etmeye özen gösterdik. Bu nedenle mimarileri kullanırken belirli eğitim aralıklarında test veri seti ile eğitilen modeli kullanarak çıktılar elde ettik. Elde ettiğimiz çıktılar arasından mimariler için en iyi çıktıları göz önünde bulundurarak değerlendirmemizi gerçekleştirdik.

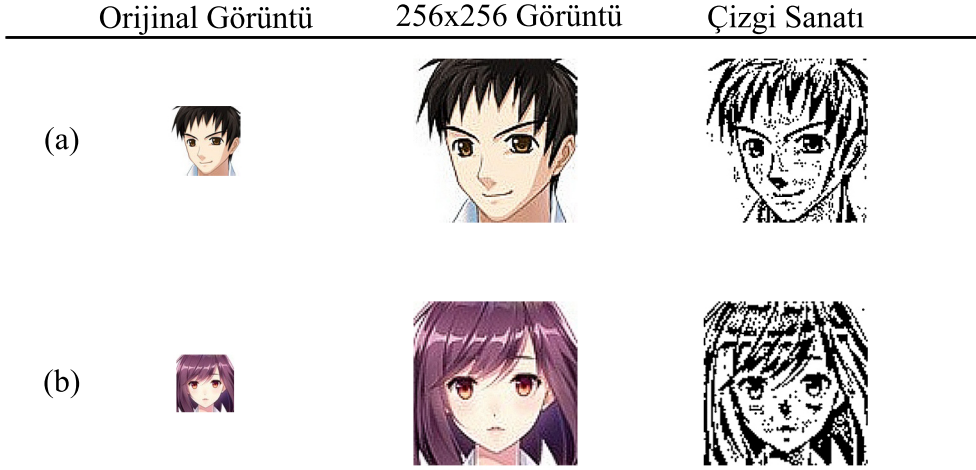
3.1.1 Kullanılacak Veri Setinin Hazırlanması

Deneylerimiz için Anime yüz veri setini (AYV) kullandık [2]. Anime yüz veri setinden (Anime Face Dataset (AFC)) toplam 21.000 görüntüyü kullanmak üzere aldık. Aldığımız görüntülerden 20.000 görüntüyü eğitim için, 1.000 görüntüyü ise test için kullandık.

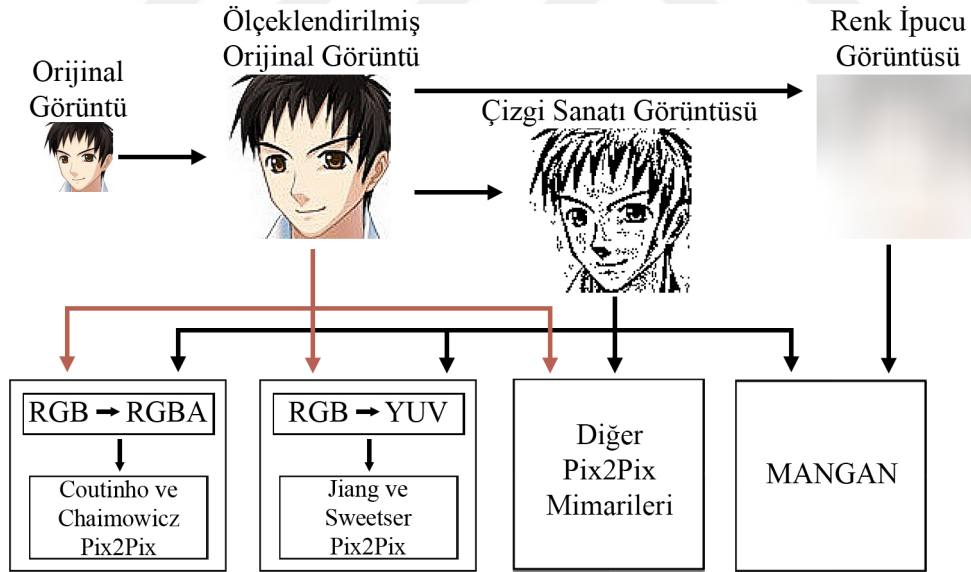
Şekil 3.2.a ve Şekil 3.2.b veri kümesinden iki örneği göstermektedir.

Veri kümesini işlemeyen önce, tüm görüntüleri 256x256 boyutuna ölçeklendirdik.

Kullandığımız anime yüz veri setindeki her görüntü için çizgi sanatına ihtiyacımız vardı, ancak anime yüz veri seti çizgi sanatı görüntüleri içermiyordu. Bu nedenle, orijinal görüntülerden çizgi sanatı oluşturmak için çeşitli yöntemler araştırdık. MANGAN [43] mimarisinde kullanılan ve amaçlarımız için gerekli olan çizgi sanatı görüntüyü üretmemizi sağlayan uyarlanabilir eşik yöntemini seçtik (Şekil 3.2, Bölüm 2.2.3). Oluşturulan çizgi sanatı görüntülerini deneylerimiz için girdi olarak kullandık. Gelecekteki çalışmalar, daha iyi çıktılar elde etmek için çizgi sanatı üretme yöntemlerini geliştirebilir veya değiştirebilir.



Şekil 3.2 Yeniden ölçeklendirilen ve üretilen çizgi sanatı görüntüleri. (a) Veri setinin orta sıralarında bulunan örnek görüntü. (b) Veri setinin sonlarına doğru örnek görüntü.

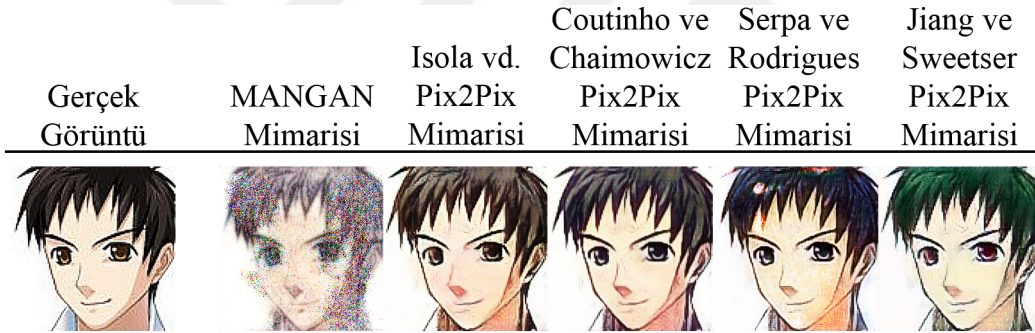


Şekil 3.3 Kıyaslama yaparken yaptığımız işlemlerin kısa bir özetini içeren şema. Mimarilerin girdilerinin daha iyi anlaşılmasını sağlayacaktır. Jiang ve Sweetser [52] tarafından ortaya atılan Pix2Pix mimarisini kullanırken yaptıkları çalışmadaki gibi, YUV renk uzayının Y kanalından çizgi sanatı üretme yönteminden yararlanmadık. Bunun yerine ürettiğimiz çizgi sanatını YUV renk uzayına çevirerek kullandık. Bunu deneylerimizde girdi olarak büyük farklılıkların önüne geçmek için yaptık.

3.1.2 Mimarilerin Karşılaştırılması

Deneylerimizde kullanacağımız mimarilerin girişleri için ayarladığımız çizgi sanatı görüntüleri ile beraber gereken girdileri karşılar hale geldik (Şekil 3.3). Temelde Pix2Pix mimarilerini kullanan çalışmalar için giriş verisi olarak orijinal görüntü ile beraber elde ettiğimiz çizgi sanatı çizimlerini kullandık. MANGAN [43] mimarisi için ise elde ettiğimiz çizgi sanatı görüntülerinin yanı sıra orijinal çalışmada belirtilen renk ipucu görüntüsünü mimariye girdi olarak verdik. Tüm girdileri ayarladıktan sonra mimarilerin oluşturduğu en iyi çıktı görüntülerini paylaştık.

Mimarilerden elde ettiğimiz sonuçlara göre karakter renklendirme işleminin otomatikleştirilmesi için temellerin atıldığını düşünüyoruz (Şekil 3.4). Ancak orijinal görüntüye benzerlik anlamında sonuçlara bakıldığında görülebileceği üzere, tam anlamıyla istenen kalitede bir sonuç elde edilememektedir. Özellikle gölgelendirme yapılan alanlarda sorun yaşanmaktadır. Bunun yanı sıra girdi olarak verilen orijinal görüntü ile birebir aynı renkler ve görüntü kalitesi elde edilememektedir. Orijinal görüntüye yakınsama anlamında sorunlar yaşanmaktadır.



Şekil 3.4 Mimarilerin orijinal görüntüye benzerlik anlamında ürettiği en iyi çıktılar. OKH değerlendirme yöntemi baz alınmıştır (Tablo 3.1).

Tablo 3.1 Mimarilere ait OKH skorları. MANGAN [43] mimarisinin ürettiği görüntüler çok gürültülü olduğu için dahil edilmemiştir.

Mimari	Şekil 3.4'deki Örnek Görüntü OKH Skoru	4 Farklı Görüntü İçin OKH Skor Ortalaması
Isola vd. [45] Pix2Pix	71,56	83,38
Coutinho ve Chaimowicz [55] Pix2Pix	71,15	83,74
Serpa ve Rodrigues [49] Pix2Pix	71,25	83,92
Jiang ve Sweetser [52] Pix2Pix	73,16	83,87

Sonuçları nesnel bir inceleme sunmak amacıyla ortalama karesel hata (mean squared error (MSE)) yöntemiyle inceledik. En iyi OKH (ortalama karesel hata) sonuçlarına sahip üretilen görüntüleri Şekil 3.4'de paylaştık. Örnek bir test görüntüsünün yanı sıra 4 farklı test görüntüsü için ortalama OKH değerlerini de

hesapladık. Tablo 3.1’de OKH değerlerini paylaştık. OKH değeri bizim için önemli olmakla beraber, OKH değeri bozuk üretilen görüntüler için hatalı değerler ortaya koyabilmektedir. Daha iyi bir ölçüt olabilme ihtimalini göz önüne alarak OKH ile değerlendirmenin yanı sıra ortalama mutlak hata (mean absolute error (MAE)) yöntemi ile değerlendirmeyi de denedik. Ancak OMH (ortalama mutlak hata) ile değerlendirmelerimizde en az hataya sahip görüntüler eğitimin başında üretilen ilk görüntüler olarak belirlendi. Bu nedenle çalışmamız boyunca OKH değerleri üzerinden ilerlemeye karar verdik.

Tablo 3.1’de görülebileceği üzere örnek görüntü için en iyi skoru alan mimari Coutinho ve Chaimowicz [55] tarafından ortaya konan Pix2Pix mimarisidir. Ancak ortalama OKH olarak Isola vd. [45] Pix2Pix mimarisi daha iyi sonuç almıştır. Aynı zamanda Serpa ve Rodrigues [49] Pix2Pix mimarisi de örnek görüntü için, Isola vd. [45] Pix2Pix mimarisinden daha iyi sonuç almıştır. Ancak üretilen görüntüye Şekil 3.4’de baktığımızda görüntüde kısmi bozukluklar bulunmaktadır. Aynı zamanda mimarinin ürettiği diğer görüntülerde de bozukluklar meydana gelmiştir. Serpa ve Rodrigues [49] Pix2Pix mimarisinin ürettiği görüntülerdeki bozuklukları dikkate alarak, aynı zamanda 4 farklı test görüntüsü için ortalama OKH değeri olarak daha iyi sonuçlar üretmesini göz önüne alarak Isola vd. [45] Pix2Pix mimarisini en iyi mimari olarak belirledik. Deneylerimize en iyi mimari olarak belirlediğimiz Isola vd. [45] Pix2Pix mimarisi ile devam ettik.



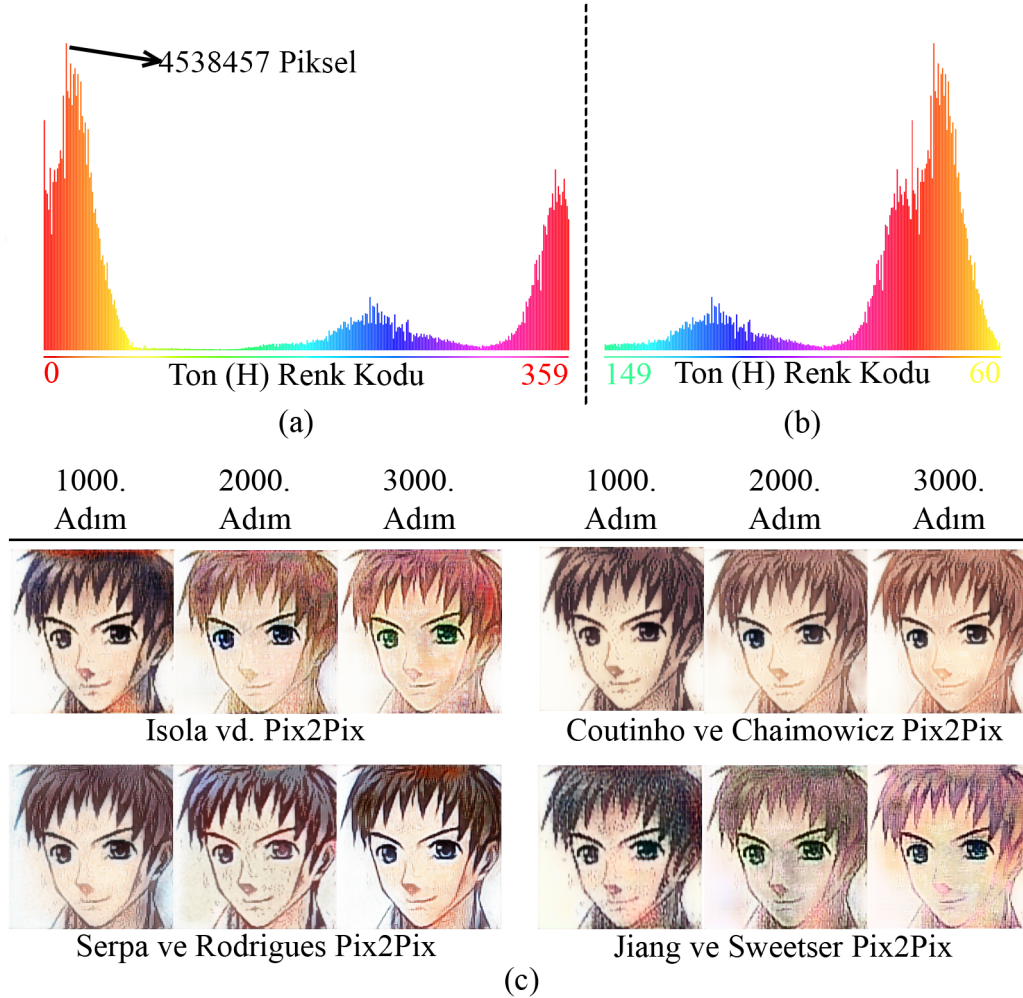
Şekil 3.5 Isola vd. [45] tarafından ortaya konan Pix2Pix mimarisinin orijinal görüntüye yakın bir görüntü ürettikten sonra, ilerleyen adımlarda ürettiği görüntüler.

Mimarileri kullanarak ürettikleri çıktıları görsel olarak değerlendirdik. İncelememizin sonuçlarını paylaşırken deneylerimize devam etmeye karar verdiğimiz Isola vd. [45] Pix2Pix mimarisini kullandık. İncelemeyi gerçekleştirmemizin amacı, üretilen görüntülerin orijinal görüntülere yaklaştıktan sonraki aşamalarında oluşan değişimlerini değerlendirmektir. İncelememiz sonucunda görsel olarak orijinal görüntüye yaklaşılacak alanlarda görüntünün genel anlamda orijinal görüntüye benzer olmaktan çok çabuk saptığını fark ettik. Daha iyi anlaşılabilmesi için Şekil 3.5’te orijinal Pix2Pix mimarisinin ürettiği görüntülerden örnek çıktıları paylaştık.

3.1.3 Sorunların Tanımlanması

Orijinale benzer üretilen görüntü çeşitliliğinin ve üretilen görüntü kalitesinin sağlanmasının önüne geçen durumları görebilmek amacıyla eğitimin farklı aşamalarında teste soktuğumuz mimarilerin ürettikleri görüntüleri inceledik. İncelemelerimiz sonucunda 2 farklı konu başlığını ele aldık.

3.1.3.1 Veri Setinin Renk Dağılımı



Şekil 3.6 (a) Eğitim verisinin HSV renk uzayında ton parametresine ait histogramı. (b) HSV renk uzayında doygunluk ve değer parametrelerini göz ardı ettiğimizde ton parametresinin bir döngü içerisinde olduğu bilinmektedir. Ton (H) parametresi 360 değeri ile beraber aslında başa dönmektedir. Histogramın daha kolay okunabilmesi amacıyla okunabilirliğe göre düzenleme yaptık ve çok az pikselde bulunan ton parametresinin değerlerini histogramdan kaldırdık. (c) Eğitimin başlangıcında üretilen görüntüler. Renk baskınlığının eğitimin ilk adımlarındaki etkisini ortaya koymak için paylaştık.

İlk sorun olarak eğitimin ilk adımlarından başlayarak ilerleyen birçok adımda renklendirme işleminin belirli renk tonlarında kaldığını fark ettik. Belirli renk tonlarında renklendirme işleminin sürdürülmesinin orijinal görüntüye benzerlik

durumu için sorun olabileceğini düşünüyoruz. Düşüncemizin ve incelememizin doğruluğundan emin olmak amacıyla eğitim veri setinin, HSV renk uzayında renk tonlarının histogramını incelemeye karar verdik [87].

Eğitim veri setimizi HSV renk uzayında incelerken siyah renk tonlarının ve beyaz renk tonlarının bu renklendirme tonlamasına etki etmemesi için eşik değeri belirledik. Eşik değeri belirleyerek görüntünün arka planında bulunan beyaz renk tonlarının ve karakterin çiziminde kullanılan siyah renk tonlarının baskınlığını elemeyi amaçladık. Sonuç olarak elde ettiğimiz histograma baktığımızda, düşündüğümüz gibi eğitim veri setinde baskın olan renk tonlarının özellikle eğitimin ilk adımlarında görüntü içerisinde de baskın olduğunu gösterdik (Şekil 3.6). Baskınlığın önüne geçilebilmesi için gelecekte yapılacak çalışmalarda renk tonları açısından daha iyi dağılıma sahip bir veri seti kullanılabileceğini düşünüyoruz. Aynı zamanda yaptığımız inceleme sayesinde renklendirilmek istenen karakterin belirli bir renk teması içerisinde renklendirilmesi istendiğinde, kullanılacak veri seti ile bir renk paleti sağlanabileceğini gösterdik.

Yaptığımız HSV renk uzayı incelemesi için HSV renk uzayının detaylarını açıklıyoruz [88]. HSV renk uzayı içerisinde ton (hue(H)), doygunluk (saturation(S)) ve değer (value(V)) parametrelerini bulunduran bir renk uzayıdır. Ton parametresi rengin tonuna biçilen değeri tutar. Doymunluk parametresi rengin doymunluğunu ve değeri parametresi ise renge ait parlaklık değerini belirtir. Siyah ve beyaz tonlarını elimine etmek amacıyla doymunluk ve değeri parametrelerinde eşik belirledik. Eşik değerimize HSV renk uzayındaki renkleri inceleyerek karar verdik. İncelememiz sonucunda doymunluk ve değeri parametrelerinin yüzde 40 değerinin altında kaldıklarında renk tonlarının beyaza veya siyaha kaydığını düşünüyoruz. Yaptığımız denemeler sonucunda eşik değerimizi yüzde 40 olarak belirledik. Belirlediğimiz eşik değerini koşul olarak kullanarak veri setimizdeki tüm görüntülerin her bir pikselini ait oldukları renk tonuna yani ton parametresine ekledik.

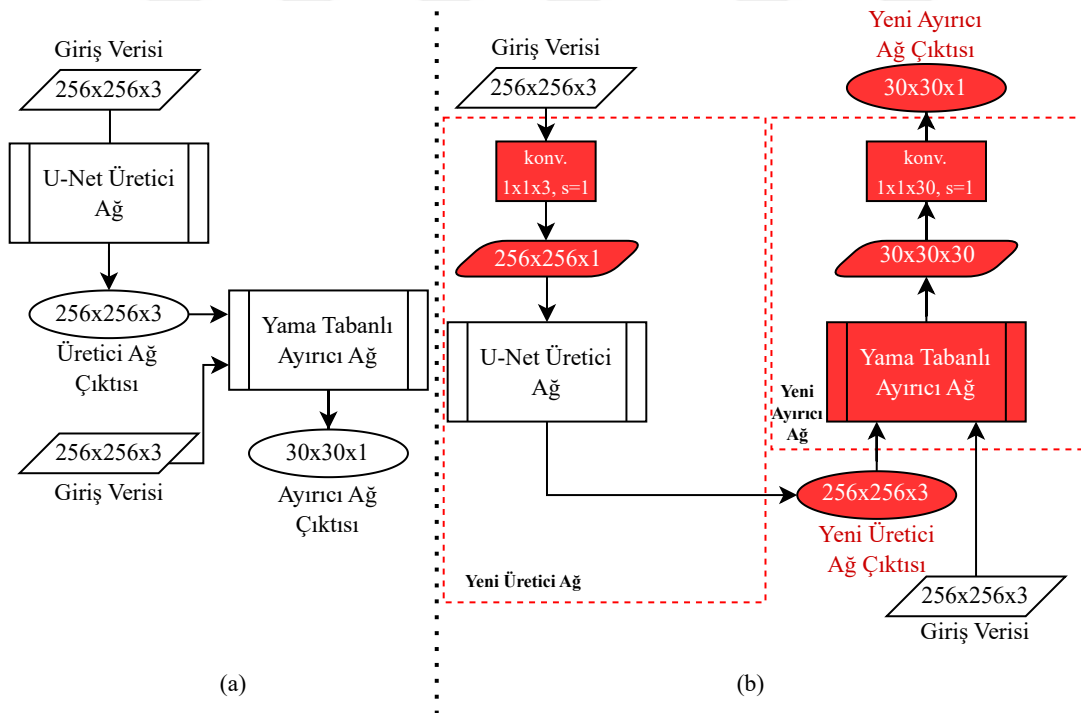
3.1.3.2 Görüntünün Bir Bütün Olarak Renklendirilmesi

Çıktıları incelemeye devam ederken göze çarpan 2. bir sorunla daha karşılaştık. Elde ettiğimiz çıktıları incelemeye devam ettiğimizde üretilen görüntülerin önemli bir çoğunluğunun bir bütün olarak renk değıştirdiğini gördük. Aynı zamanda renklendirme işlemi yapılırken kullandığımız mimarilerin karakterin tamamına yönelik boyama yaptığını fark ettik. Şekil 3.5'te paylaştığımız Isola vd. [45] tarafından ortaya konan Pix2Pix mimarisinin ürettiği görüntüler incelendiğinde de görüntünün bir bütün olarak renklendirildiği görülmektedir.

Karakterin tamamına yönelik boyama yapılmasının dolaylı bir şekilde mimariler tarafından üretilen görüntülerde kaliteyi düşürdüğünü düşünüyoruz. Yapılan renklendirmelerin bir bütün olarak yapılması, minimal değişiklikleri sağlayamamak anlamına gelmektedir. Minimal değişikliklerin sağlanamaması üretilen görüntünün orijinale yaklaşması durumunda, minimal değişikliklerle kalitenin ve orijinal görüntüye yakınlığın artmasının önüne geçmektedir. Özellikle orijinal görüntüye yakın bir görüntü oluşturulan adımdan sonra, mimarinin görüntünün renklerini bir bütün olarak değiştirmeye yönelmesinin, orijinal görüntüye daha yakın sonuçlar üretilmemesine neden olduğunu düşünüyoruz. Tez çalışmamızda kaliteyi, orijinale yakınlığı ve üretilen görüntülerde minimal değişiklikleri sağlayarak orijinale yakın görüntü çeşitliliğini de artırabilmek amacıyla bu sorunun üzerine gittik.

3.1.4 İyileştirmeler

Görüntünün bir bütün olarak boyanması problemini çözmek için bazı denemeler yaptık. Deneylerimizi yaparken Isola vd. [45] Pix2Pix mimarisini kullandık (Şekil 3.7.a). Isola vd. [45] tarafından ortaya konan orijinal Pix2Pix mimarisini kullanmamızın nedeni, Bölüm 3.1.2’de belirttiğimiz gibi en iyi sonuçları aldığımız mimari olmasından kaynaklanmaktadır.



Şekil 3.7 (a) Orijinal Pix2Pix Mimarisi [45]. (b) Ortaya koyduğumuz Pix2Pix modifikasyonu (Line2Pix). Bizim eklediğimiz katmanları kırmızı renkle renklendirdik. Genel olarak kodlamayı Pix2Pix mimarisinden aldık. Mimarimizi daha iyi anlaşılabilmesi için görsel olarak ortaya koyduk. Yaptığımız eklemelerde çekirdek boyutu (kernel size) olarak 1x1, adım (stride) olarak 1 değerini kullandık.

Kalite ve orijinale yakın görüntü çeşitliliğini artırmak amacıyla, aynı zamanda görüntünün bir bütün olarak renklendirilmesi problemini çözebilmek amacıyla Saravanan ve Guzdial [83] tarafından önerilen PixelSight bloklarını Pix2Pix mimarisine uyarlamaya karar verdik (Şekil 3.7.b). Uyarlamayı yapabilmek amacıyla üretici ağ girişine ve ayırıcı ağ çıkışına 1x1 evrişim katmanları (konvolüsyonlar) ekledik. Üretici ağ girişine eklediğimiz 1x1 evrişim katmanı ile beraber model karmaşıklığını azaltmayı hedefledik. Aynı zamanda giriş verimizde kenar kısımları önemli olduğu için, giriş verimizdeki gereksiz olabilecek özelliklerden uzaklaşmayı amaçladık. Mimarinin giriş görüntüsünü yorumlayarak kullanmasını sağladık.

Ayırıcı ağ için Isola vd. [45] tarafından ortaya konan orijinal Pix2Pix mimarisinde 70x70 algılama alanı (receptive field) kullanılmıştır. Isola vd. [45] Pix2Pix makalesinde de belirtildiği üzere farklı algılama alanları denenerek 70x70 tercih edilmiştir. Ayırıcı ağ çıkış katmanında ise 30x30x1 çıktısı verilmektedir. Mimaride değişikliğe giderek çıkış katmanını 30x30x30 olacak şekilde güncelledik. Elde ettiğimiz 30x30x30 çıktısına 1x1x30 evrişim uygulayarak 30x30x1 çıktısını elde ettik. Kısaca orijinal Pix2Pix mimarisinin çıktı aldığı katmanda, 30 farklı değerlendirme sağladık ve daha sonra sağladığımız değerlendirmelerden tek bir sonuç elde etmeye odaklandık. Yaptığımız ekleme sayesinde 70x70 algılama alanını değiştirmeden, mimarinin daha fazla değerlendirme yapmasına olanak sağladık.



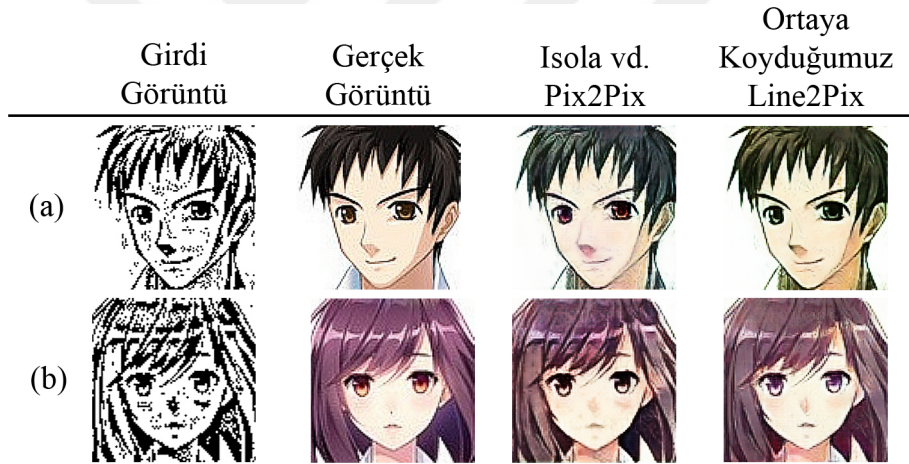
Şekil 3.8 Mimarilerin eğitimin başlangıç aşamalarındaki renk değişimleri. Ortaya koyduğumuz Line2Pix mimarisinin renklendirmeye başlamak için ihtiyaç duyduğu eğitim süresi daha fazladır. Minimal değişikliklere izin vermemiz sonucu eğitim süreleri uzamıştır.

Yaptığımız değişimlerle beraber daha tutarlı bir öğrenmenin önünü açtık. Ortaya koyduğumuz Line2Pix mimarisi ile daha fazla değerlendirme yaparak görüntünün daha iyi değerlendirilmesinin önünü açtık. Daha iyi değerlendirme sağlayarak, mimarinin minimal değişimlere yönelmesini sağladık.

Line2Pix mimarisi ile beraber kaliteyi ve orijinale benzer anlamda üretilen görüntü çeşitliliğini artırdık. Orijinal görüntüye yakın bir görüntü üretildikten sonra,



Şekil 3.9 Yaptığımız iyileştirmeler sonucunda orijinal görüntüye yaklaşılın üretim adımından sonraki adımlarda üretilen örnek görüntüler.



Şekil 3.10 Yaptığımız iyileştirmeler sonucunda 2 farklı görüntümüz için OKH yöntemine göre üretilmiş en iyi görüntüler. Görüntülere ait OKH skorları Tablo 3.2’de paylaşılmıştır. (a) Orijinal Pix2Pix mimarisi ile üretilen görüntüde yüz bölgesinde mavi renk tonları bulunmaktadır. (b) Line2Pix mimarimiz ile üretilen görüntünün hem yüz rengi, hem de saç rengi orijinal görüntüye daha yakındır.

Tablo 3.2 Şekil 3.10’da paylaştığımız görüntülere ait OKH skorları. Aynı zamanda 4 farklı test görüntümüz için aldığımız OKH skorlarının ortalamasını paylaştık.

Mimari	Şekil 3.10.a’daki Örnek Görüntü OKH Skoru	Şekil 3.10.b’deki Örnek Görüntü OKH Skoru	4 Farklı Görüntü İçin OKH Skor Ortalaması
Isola vd. [45] Pix2Pix	69,92	87,67	82,55
Ortaya Koyduğumuz Line2Pix	66,07	83,71	80,2

ilerleyen adımlarda daha uzun süre boyunca minimal değişikliklerle orijinale yaklaşılmmasını sağladık (Şekil 3.9). Değerlendirme metodu olarak OKH yöntemini kullanmaya devam ettik. Orijinale yakınlık olarak OKH yöntemiyle en iyi skoru alan görüntüleri Şekil 3.10'da paylaştık. OKH yöntemi ile elde ettiğimiz sonuçları Tablo 3.2'de paylaştık. OKH yönteminde sorunlar yaşanabildiği için çıktıları aynı zamanda gözle de kontrol ettik. Isola vd. ([45]) Pix2Pix mimarisine göre daha uzun süren eğitimlerle daha iyi sonuçlar aldık. Yaptığımız değişikliklerle adım sayısı aynı kalsa da, eklediğimiz yeni katmanlar nedeniyle eğitim süreci daha yavaş ilerlemektedir. Kısacası eğitimde adım başına düşen süre uzamıştır. Minimal değişikliklerin önünü açarak, daha istikrarlı ve daha tutarlı bir öğrenme sağladığımız için Isola vd. [45] Pix2Pix mimarisine göre daha uzun eğitim sürelerine ihtiyaç duyduk (Şekil 3.8).

Mimarimiz orijinal Pix2Pix mimarisi tabanlıdır ve aynı maliyet fonksiyonunu (Bölüm 2.2.4) kullanmaktadır (Şekil 3.7.b). Adam yöntemi (Bölüm 2.1) ile optimizasyon sağlanmıştır. Öğrenme katsayısı (learning rate) değeri olarak $2e - 4$ değeri kullanılmıştır. Diğer hiper parametreler için $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 1e - 07$ değerleri kullanılmıştır.

3.1.5 Gerçek Dünya Verileriyle Test Süreci

Karakter renklendirmesi ile ilgili son deneyimizde gerçek dünya verileri üzerinde denemeler gerçekleştirdik. Gerçek dünya verileri ile çalışmaları yaparken, mimarilerin test sürecinde girdi olarak gerçek dünya verilerine yakın çizimlerle çalıştık. Ancak mimarilerin eğitim süreçlerinde, elimizde çok fazla gerçek dünya verilerine yakın çizim ve çıktı olmadığı için farklı yöntemlere başvurduk.

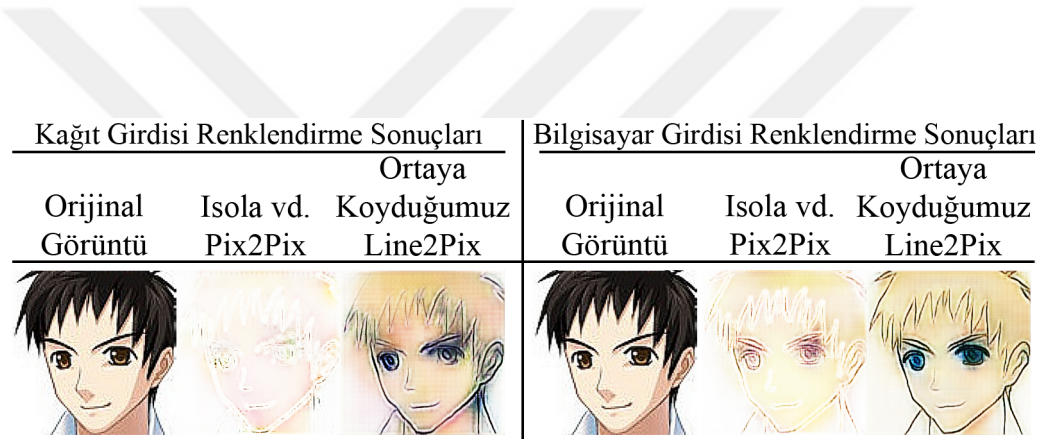
3.1.5.1 Uyarlanabilir Eşik Yöntemi ile Eğitim

Gerçek dünya verileri ile test sürecimizde ilk olarak mimarileri eğitirken orijinal görüntülerden uyarlanabilir eşik yöntemi ile elde ettiğimiz çizgi sanatı görüntüleri kullandık. Uyarlanabilir eşik yöntemi ile elde ettiğimiz çizgi sanatı görüntülerle eğittiğimiz mimarileri, kendi el çizimlerimiz ile oluşturduğumuz görüntüler ile test ettik. El çizimlerimizi yaparken uyarlanabilir eşik yöntemi ile elde ettiğimiz çizimlere benzemesine özen gösterdik (Şekil 3.11).

El çizimlerimize ait görüntünün bilgisayara aktarıldığında soluk kaldığını fark ettik. Bu nedenle, daha iyi sonuçlar üretebileceğini düşündüğümüz için bilgisayar ortamında çizimi yeniden yapılandırdık (Şekil 3.11). Yeniden yapılandırılmış çizimle beraber mimarilerimizi test ettik (Şekil 3.12).



Şekil 3.11 Bize ait olan el çizimleri.



Şekil 3.12 Bize ait olan el çizimlerini (Şekil 3.11) test verisi olarak kullandığımızda ürettiğimiz sonuçlar. Girdi olarak kullandığımız kağıt ve bilgisayar çizimleri Şekil 3.11’de görülebilir. Kağıt girdisi ile üretilen sonuçlara ait OKH skorlarını Tablo 3.3’te, bilgisayar girdisine ait OKH skorlarını Tablo 3.4’te paylaştık. Mimarilerin eğitiminde uyarlanabilir eşik yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.

Tablo 3.3 Mimarilere kağıt girdisi (Şekil 3.11) verilerek üretilen görüntülere ait OKH skorları. Mimarilerin eğitiminde uyarlanabilir eşik yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.

Mimari	Şekil 3.12’de Paylaşılan Görüntüye Ait OKH Skoru	4 Farklı Görüntü İçin Ortalama OKH Skoru
Isola vd. [45] Pix2Pix	77,62	86,48
Ortaya Koyduğumuz Line2Pix	99,3	100,13

Tablo 3.4 Mimarilere bilgisayar girdisi (Şekil 3.11) verilerek üretilen görüntülere ait OKH skorları. Mimarilerin eğitiminde uyarlanabilir eşik yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.

Mimari	Şekil 3.12’de Paylaşılan Görüntüye Ait OKH Skoru	4 Farklı Görüntü İçin Ortalama OKH Skoru
Isola vd. [45] Pix2Pix	79,17	87,97
Ortaya Koyduğumuz Line2Pix	93,76	97,32

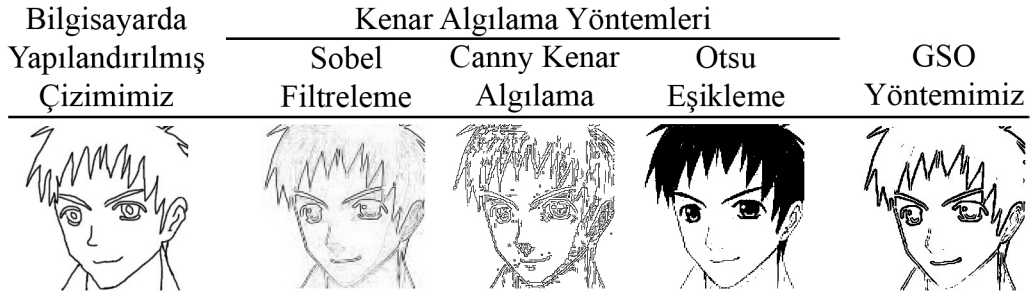
Yaptığımız deney sonuçları üretilen görüntüleri OKH yöntemi ile değerlendirdiğimizde orijinal Pix2Pix mimarisinin daha iyi sonuçlar aldığını gördük (Tablo 3.3, Tablo 3.4). Ancak paylaştığımız Şekil 3.12’de de görüldüğü üzere ortaya koyduğumuz Line2Pix mimarisinin renklendirme işlemini daha iyi bir şekilde yaptığı gözükmemektedir. Isola vd. [45] Pix2Pix mimarisi görüntünün yapısını bozarken, ortaya koyduğumuz Line2Pix mimarisi görüntünün yapısını bozmamaktadır. Dolayısıyla yaptığımız iyileştirmelerin etkisi de yaptığımız deney ile daha net bir şekilde görülebilmektedir.

Ortaya koyduğumuz Line2Pix mimarisi renklendirme işlemini yapmaya çalışsa da kaliteli bir şekilde sağlayamamıştır. Ancak görüntüyü renklendirmeye odaklanmış ve gerekli yerlerin renk değişimlerini sağlamaya çalışmıştır. Eğitim verilerindeki çizgi sanatı görüntüsü ile test sürecindeki çizgi sanatı görüntüsü farklı olduğu için sonuçların kalitesiz olmasının normal olduğunu düşünüyoruz.

3.1.5.2 Yeni Çizgi Sanatı Görüntüsü Oluşturma Yöntemi

Gerçek dünya verileri ile çalışırken kötü sonuçlar elde ettiğimiz için, el çizimi ile ortaya koyduğumuz çizgi sanatı görüntülerine yakın görüntüler elde edebileceğimiz bir yonteme ihtiyaç duyduk. İhtiyacımız doğrultusunda mimarileri eğitirken, uyarlanabilir eşik yöntemi ile elde ettiğimiz çizgi sanatı görüntüler yerine el çizimi ile ortaya koyduğumuz çizgi sanatına daha benzer çizgi sanatı görüntüler elde etmek için denemeler yaptık.

Ortaya koyduğumuz GSO yöntemi ile orijinal görüntüye önce bulanıklık (blur) filtresini uyguladık. Ardından Sobel filtresi ile kenarları çıkarttık. Bulanıklık filtresi ile daha iyi sonuçlar elde ettiğimiz için bulanıklık filtresini kullandık. Son olarak elde ettiğimiz görüntüyü Otsu yöntemi ile filtreledik ve el çizimlerimize oldukça benzer bir çizgi sanatı görüntü çıktısını elde ettik (Şekil 3.13). Ortaya koyduğumuz GSO yöntemi ile bazı kenar algılama yöntemlerinin karşılaştırmasını sunduk (Şekil 3.13).



Şekil 3.13 Yeni çizgi sanatı görüntüsü oluşturma yöntemimiz ve kenar algılama yöntemleri.

3.1.5.3 Yeni Çizgi Sanatı Görüntüsü Oluşturma Yöntemi ile Eğitim

Ortaya koyduğumuz GSO yöntemi ile tüm veri setine ait çizgi sanatı görüntülerini oluşturduk. Oluşturduğumuz yeni çizgi sanatı görüntülerle mimarileri baştan eğittik. Eğittiğimiz mimariler ile kağıt üzerindeki ve bilgisayar ortamında yapılandırdığımız el çizimlerimizi kullanarak renklendirme için denemeler yaptık. OKH yöntemine göre elde ettiğimiz en iyi çıktıları Şekil 3.14'te paylaştık. Kağıt girdisi ile elde ettiğimiz OKH skorlarını Tablo 3.5'te, bilgisayar girdisi ile elde ettiğimiz OKH skorlarını Tablo 3.6'da paylaştık. Yeni çizgi sanatı görüntüsü oluşturma yöntemimiz ile beraber mimarilerin ürettikleri görüntülerin daha iyi olduğu görülmektedir.



Şekil 3.14 Mimarileri yeni çizgi sanatı oluşturma yöntemimizle elde ettiğimiz çizgi sanatı görüntülerle eğittikten sonra, kağıt ve bilgisayar çizimlerini girdi olarak verdiğimizde mimarilerden elde ettiğimiz görüntüler. Girdi olarak kullandığımız kağıt ve bilgisayar çizimleri Şekil 3.11'de görülebilir.

Tablo 3.5 Mimarilere kağıt girdisi (Şekil 3.11) verilerek üretilen görüntülere ait OKH skorları. Mimarilerin eğitiminde yeni çizgi sanatı oluşturma yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.

Mimari	Şekil 3.14'te Paylaşılan Görüntüye Ait OKH Skoru	4 Farklı Görüntü İçin Ortalama OKH Skoru
Isola vd. [45] Pix2Pix	90,32	94,83
Ortaya Koyduğumuz Line2Pix	87,19	93,75

Tablo 3.6 Mimarilere bilgisayar girdisi (Şekil 3.11) verilerek üretilen görüntülere ait OKH skorları. Mimarilerin eğitiminde yeni çizgi sanatı oluşturma yöntemiyle elde edilen çizgi sanatı görüntüler kullanılmıştır.

Mimari	Şekil 3.14'te Paylaşılan Görüntüye Ait OKH Skoru	4 Farklı Görüntü İçin Ortalama OKH Skoru
Isola vd. [45] Pix2Pix	88,92	93,86
Ortaya Koyduğumuz Line2Pix	87,85	93,59

3.2 Karakter Üretimi

Sıfırdan karakter üretimi otomasyonu, oyun ve çizgi romanlar için oldukça önemli bir adım olma potansiyeline sahiptir. Karakter renklendirme sürecinde olduğu gibi karakter üretimi de tasarımcıya fikir verme açısından kullanılabilir. Bu bölümde tez çalışmamızda Bölüm 1.1.1 bahsettiğimiz piksel sanatı tarzındaki karakterlerle deneyi gerçekleştirdik.

Kullandığımız veri setinde farklı yöne bakan karakterler bulunmaktadır [89]. Veri setinde bulunan verilerin sadece yüzü ekrana dönük olanlarını kullandık.

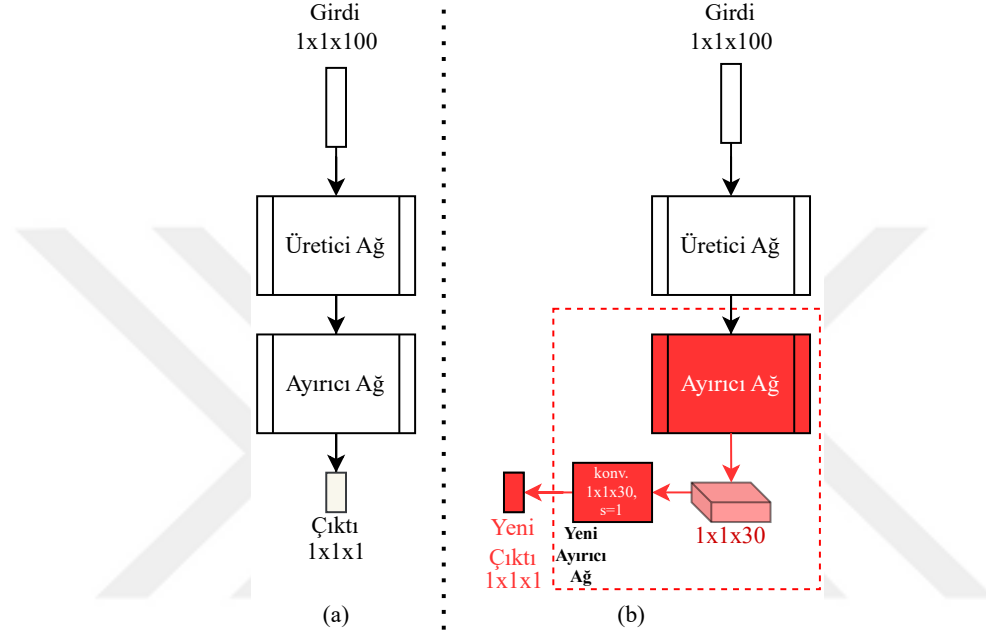
3.2.1 Deney Süreci ve İyileştirmeler

Deneylerimizde, veri setimizle eğitilmiş derin konvolüsyonel çekişmeli üretici ağ (Bölüm 2.2.2) mimarisini kullandık (Şekil 3.15.a). Horsley ve Perez-Liebana [41] tarafından yapılan çalışmadan esinlenerek DKÇÜA mimarisini karakter üretiminde kullanmaya karar verdik. Inkawich [90] tarafından oluşturulan DKÇÜA mimarisini temel alarak deneylerimizi gerçekleştirdik. Kullanmakta olduğumuz derin konvolüsyonel çekişmeli üretici ağ mimarisinde, oluşturulan görüntüleri paylaşmadan önce birçok kez eğitim denemeleri gerçekleştirdik. Eğitim denemelerimiz sırasında zaman zaman DKÇÜA mimarisinde sapmalarla karşılaştık ve bu da gürültülü görüntülere yol açtı. Sapma sorununu çözmek için DKÇÜA mimarisine tarihsel ortalama yönteminden (historical averaging method) esinlenen bir değişiklik önerdik [91].

$$\theta_{t+1} = (1 - \alpha)\theta_t + \alpha\theta'_{t+1} \quad (3.1)$$

Ortaya koyduğumuz yöntem Denklem 3.1'de gösterilmiştir. θ_{t+1} geçmiş ağırlıklarla yeni ağırlıkların işleme girmesi sonucu elde edilen yeni ağırlıklardır. θ_t geçmiş ağırlıkları temsil eder. θ'_t ise hesaplamalar sonrası yeni adımda elde edilen, geçmiş ağırlıklarla henüz işleme girmemiş ağırlıkları temsil eden ifadedir. α hiper

parametresi henüz hesaplamada kullanılmayan ağırlık değerlerinin (θ'_t) önemini belirtir. Çalışmamızda $\alpha = 0.3$ değerini kullanıyoruz. Uyguladığımız yöntem sayesinde geçmiş ağırlıklara bağlı olarak ağırlık güncellemesi yapabiliyoruz ve sapmanın etkisini azaltmaya yardımcı olduğunu düşünüyoruz. Mimariyi sunduğumuz yöntem ile birden çok kez eğitmeye çalıştık. Görüntü üretmeye başladıktan sonra tekrar gürültü üretmedik. Dolayısıyla sapmaya rastlamadık ve bu nedenle sapmayı büyük ölçüde önlediğimize inanıyoruz.



Şekil 3.15 (a) DKÇÜA mimarisine genel bakış [90]. (b) Karakter üretimi için ortaya koyduğumuz mimarimize genel bakış.

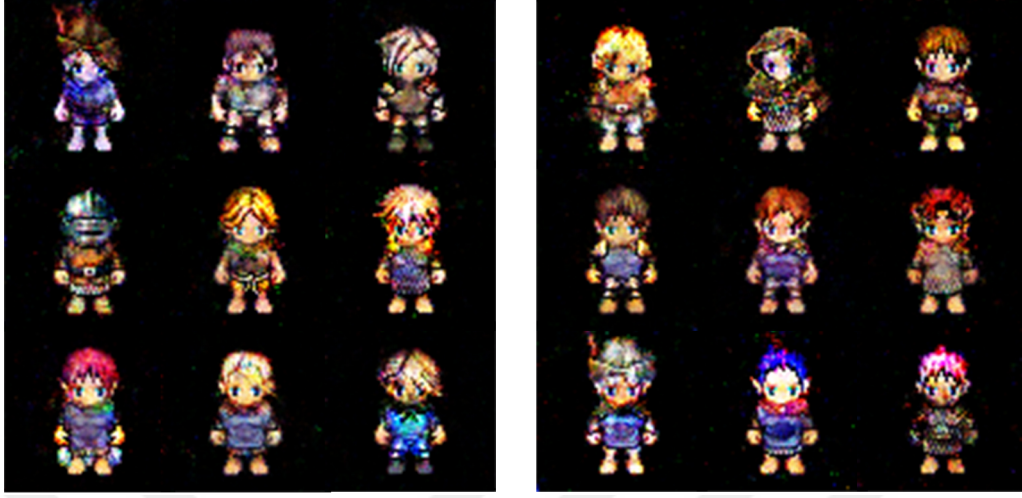
Üretilen karakterlerin kalitesini artırabileceğini düşündüğümüz için ayırıcı ağ çıkış katmanında üretilen sonucu 1x1x30 olarak güncelledik. Güncellememiz sayesinde ayırıcı ağın farklı incelemeler yapmasını sağlamaya çalıştık. 30 farklı değerlendirme sağlayan ayırıcı ağa son katman olarak 1x1 evrişim ekledik ve farklı değerlendirmeler arasında karar vermesini sağladık (Şekil 3.15.b). Çalışmamızı yaparken PixelSight bloklarından esinlendik [83]. Çalışmamızın sonuçlarını paylaştık (Şekil 3.16).

3.2.2 Kullandığımız Parametreler

Ortaya koyduğumuz DKÇÜA mimarisi ÇÜA mimarisiyle aynı maliyet fonksiyonu (Bölüm 2.2.1) üzerinden işlemlerini sürdürür. Yaptığımız değişiklikler fonksiyonlar içerisinde olduğundan dolayı maliyet fonksiyonu değişmeyecektir. Yığın boyutu (batch size) 128 olarak belirlenmiştir. Tüm resimler 64x64 olarak ölçeklendirilmiştir. Adam yöntemi ile optimizasyon sağlanmıştır ve parametreler

Derin Konvolüsyonel
Çekişmeli Üretici
Ağ (DKÇÜA)

İyileştirmelerimiz
Sonucu DKÇÜA



Şekil 3.16 Karakter üretimi için kullandığımız mimarilerden elde ettiğimiz örnek çıktılar.

için öğrenme katsayısı $= 2e - 4$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 1e - 08$ değerleri kullanılmıştır. Sapmalardan kurtulmak amacıyla eklediğimiz yöntemde $\alpha = 0.3$ değerini kullandık (Denklem 3.1).

3.2.3 Kullanıcı Aracılığı ile Değerlendirme

Eğitim sırasında, yapay zeka modelinin belirli bir veri grubunu işlemesine adım denir. Tüm veri seti model tarafından bir kez işlendiğinde ise bu sürece döngü adı verilir. Her 100 döngüde 1 çıktı aldık. Genellikle en iyi çıktıyı sağlayan 2000 ile 7000 döngü (epoch) aralığına odaklanarak değerlendirme yaptık.

Değerlendirme yöntemi olarak Fréchet başlangıç mesafesi (Fréchet Inception Distance (FID)) [92] veya başlangıç skoru (Inception Score (IS)) [91] yöntemlerini tercih etmedik. Fréchet başlangıç mesafesi (FBM) yöntemini tercih etmeme nedenimiz çeşitlilik arttıkça kötü skor vermesinden kaynaklanmaktadır. Diğer bir yöntem olan başlangıç skoru (BS), çeşitliliği dikkate alsa da, karakter üretimindeki sanatsallığı dikkate alamaz. Tüm değerlendirme yöntemlerini incelememekle beraber karakter üretimini değerlendirmede kullanılabilecek ve standart oluşturabilecek bir değerlendirme yöntemine ihtiyaç olduğunu düşünüyoruz.

Elde ettiğimiz çıktıların daha kaliteli ve çeşitli olduğuna inanıyoruz. FBM ve BS değerlendirme yöntemleri hakkındaki fikirlerimizden dolayı, elde ettiğimiz çıktıları

kullanıcı anketi aracılığıyla değerlendirdik. Anketi hazırlarken mimarilerin en iyi çıktıları ürettiği aralık olan 2000-7000 döngü aralığındaki çıktılarını kullandık. Anket için mimarilerin 2000-7000 döngü aralığında ürettiği en yüksek kaliteli ve en çeşitli çıktılarını kullandık. Kullanıcılardan görüntü kalitesi, ana hat belirginliği ve renk uyumuna göre seçim yapmalarını istedik. Son olarak, kullanıcılardan tercih ettikleri karakteri seçmelerini istedik (Tablo 3.7).

Tablo 3.7 Yaptığımız ankete ait sonuçlar.

Değerlendirme Alanı	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Görüntü Kalitesi	42,75%	57,25%
Ana Hat Belirginliği	42,75%	57,25%
Renk Uyumu	42,35%	57,65%
Tercih Edilen Karakter	39,22%	60,78%
Toplam	41,67%	58,33%

Toplam 51 katılımcı ile gerçekleştirilen anketin sonuçlarını sunduk (Tablo 3.7). Anket, mimarilerden elde edilen en kaliteli ve en çeşitli 16 çıktı kullanılarak hazırlanmıştır. Toplam 30 farklı soru grubu oluşturulmuş ve her grup için 4 farklı alanda yanıt istenmiştir. Her katılımcıya 5 soru grubu gösterilmiş ve cevapları alınmıştır. Dolayısıyla değerlendirilen alana ilişkin yüzdeler 255 oy, “Toplam” satırına ilişkin yüzdeler ise 1.020 oy üzerinden hesaplanmıştır. Kullanıcıların farklı alanlarda verdikleri tüm oyları toplu olarak değerlendirdik ve değerlendirmemizin sonucunu tablodaki “Toplam” satırında gösterdik. Ankette, iyileştirmelerimiz sonucu ortaya koyduğumuz DKÇÜA mimarisinde daha iyi sonuçlar aldığımız için $\alpha = 0.3$ değerini kullandık.

Anketimizin sonuçlarını dikkate alarak mimarileri değerlendirdiğimizde olumlu kullanıcı geri bildirimleri aldık. Öğrenmeyi daha fazla değerlendirme üzerinden sağlayarak, daha kaliteli karakterler ürettiğimizi düşünüyoruz. DKÇÜA mimarisinde yaptığımız değişikliklerin, üretilen karakterleri kullanıcılar için daha çekici hale getirdiğini anket sonuçları ile gösterdik. Ortaya koyduğumuz bulguların gelecekteki araştırmalar için fikir verebileceğine inanıyoruz.

4 SONUÇ

Tez çalışmamızda, karakter renklendirme ve sıfırdan karakter üretimi için deneyler yaptık.

Karakter renklendirme süreci için MANGAN mimarisi [43] ve Pix2Pix modifikasyonlarını karşılaştırdık. Isola vd. [45] tarafından ortaya konulan orijinal Pix2Pix mimarisi, Serpa ve Rodrigues [49] tarafından ortaya konulan Pix2Pix mimarisi, Jiang ve Sweetser [52] tarafından ortaya konulan Pix2Pix mimarisi, Coutinho ve Chaimowicz [55] tarafından ortaya konulan Pix2Pix mimarisi, karakter renklendirme deneylerinde kullandığımız Pix2Pix modifikasyonlarıdır.

Yaptığımız deneyler sonucunda en iyi sonucu Isola vd. [45] tarafından ortaya konulan Pix2Pix mimarisi ile elde ediyoruz. Elde ettiğimiz sonuçları incelediğimizde bazı sorunların öne çıktığını fark ettik ve paylaştık.

Karakter renklendirme sürecinin otomasyonunu sağlamak için kullanılan mimarilerde, eğitimde kullanılacak olan veri setinin renk baskınlığının etkisine değindik. Veri setinin renk baskınlığını histogram yöntemini kullanarak gösterdik ve sonuçlara olan etkisini ortaya koyduk. Ortaya koyduğumuz renk baskınlığı problemi için çözüm yöntemi olarak daha dengeli dağılıma sahip bir veri seti kullanmayı önerdik.

Karakter renklendirme sürecinde fark ettiğimiz diğer bir sorun görüntünün bir bütün olarak renklendirilmesi olduğunu örneklerle gösterdik. Renklendirme işleminin bütün bir şekilde yapılmasının önüne geçmek ve kaliteyi artırmak amacıyla Pixel VQ-VAE mimarisinde kullanılan PixelSight bloklarını [83] Isola vd. [45] tarafından ortaya konulan orijinal Pix2Pix mimarisine entegre ettik. Üretici ağ girişine ve ayırıcı ağ çıkışına 1x1 konvolüsyonlar ekleyerek entegre işlemini gerçekleştirdik. Eklediğimiz konvolüsyonlar sayesinde sorunun önüne geçtiğimizi gösterdik.

Karakter renklendirme ile alakalı son olarak gerçek dünya verileri ile deneyler gerçekleştirdik. Gerçek dünya verilerinin doğru bir şekilde renklendirilmesinde

mimarilerin eğitiminde kullanılan çizgi sanatı görüntülerin önemini ortaya koyduk. Gerçek dünya verilerinin daha iyi işlenebilmesi için yeni bir çizgi sanatı görüntüsü oluşturma yöntemi önerdik. Önerdiğimiz yöntem ile orijinal görüntüye sırasıyla bulanıklık filtresi ve Sobel filtresi uygulayıp, Otsu yöntemi ile eşiklendirerek çizgi sanatı görüntüsü elde ettik. Eğitimde yeni çizgi sanatı görüntülerimizi kullandığımızda ürettiğimiz sonuçların daha iyi renklendirildiğini gösterdik.

Sıfırdan karakter üretimi için Horsley ve Perez-Liebana [41] tarafından yapılan çalışmadan esinlenerek, DKÇÜA mimarisi ile deneylerimizi gerçekleştirdik. Başlangıçta bazı eğitim denemelerimizde sapmalarla karşılaştık. DKÇÜA mimarisi bir süre ilerleme kaydettikten sonra tekrar gürültü üretmeye başlıyordu. Sapmaların önüne geçebilmek amacıyla tarihsel ortalama yönteminden esinlenerek, mimarideki ağırlıkları güncellerken eski ağırlıklara bağlı kaldık. Uyguladığımız yöntem ile sapmaların önüne geçtik.

DKÇÜA mimarisi ile gerçekleştirdiğimiz deneylerde elde ettiğimiz sonuçlara baktığımızda kaliteyi artırmamız gerektiğini düşündük. Kaliteyi artırmayı sağlamak amacıyla ayırıcı ağ mimarisinde değişiklikler yaptık. Ayırıcı ağ çıkışına 1x1 konvolüsyon koyarak mimarinin daha fazla değerlendirme yapmasını sağladık.

DKÇÜA mimarisi ile iyileştirmelerimiz sonucu ortaya koyduğumuz DKÇÜA mimarisi tarafından üretilen çıktıları, kullanıcı anketi aracılığıyla yeniden değerlendirdik. İyileştirmelerle ortaya koyduğumuz DKÇÜA mimarimiz tarafından üretilen karakterlerin, standart DKÇÜA mimarisi tarafından üretilenlere göre %140 daha fazla tercih edildiğini gösterdik.

- [1] M.-H. Kuo, Y.-L. Yang H.-K. Chu, “Feature-Aware Pixel Art Animation,” *Computer Graphics Forum*, Wiley Online Library, c. 35, 2016, ss. 411–420.
- [2] S. Churchill B. Chao, *Anime Face Dataset*, 2019. doi: 10 . 34740 / KAGGLE/DS/379764. erişim adresi: <https://www.kaggle.com/ds/379764> (erişim tarihi 05/09/2024).
- [3] *Fantasy Knight - Free Animated Pixelart Character*. erişim adresi: <https://aamatniekss.itch.io/fantasy-knight-free%20%5C%5C-pixelart-animated-character> (erişim tarihi 26/08/2024).
- [4] R. Zhang, S. Xiong J. Zhang, “A study of ‘hand stitching article’ for animation model design based on ‘pixel art’,” *2008 9th International Conference on Computer-Aided Industrial Design and Conceptual Design*, IEEE, 2008, ss. 880–883.
- [5] R. Yang, Y. Wang, Y. Wang M. Xu, “Application of Neural Network in Pixel Art Creation: Bi-directional Conversion between Photo and Pixel Art with GAN Base Model,” *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, IEEE, 2022, ss. 855–858.
- [6] X. Wei Y. Wang, “Research on the Application of Pixel Style in UI (User Interface) Design,” *The 6th International Conference on Arts, Design and Contemporary Education (ICADCE 2020)*, Atlantis Press, 2021, ss. 557–559.
- [7] D. Silber, *Pixel Art for Game Developers*. Taylor & Francis, 2015, isbn: 9781482252309. erişim adresi: <https://books.google.com.tr/books?id=n0zRrQEACAAJ>.
- [8] T. Zufri, D. Hilman, O. Frans ve diğ., “Research on the Application of Pixel Art in Game Character Design,” *Journal of Games, Game Art, and Gamification*, c. 7, no. 1, ss. 27–31, 2022.
- [9] V. T. S. A. L. Apolinário, “A Feature Model Proposal for Computer Games Design,” *VII Brazilian Symposium on Computer Games and Digital Entertainment November, 10-12, 2008 Belo Horizonte-MG-BRAZIL*, Citeseer, 2008, s. 54.
- [10] C. Baker, M. Schleser K. Molga, “Aesthetics of mobile media art,” *Journal of Media Practice*, c. 10, no. 2-3, ss. 101–122, 2009.
- [11] C. Marín-Lora, M. Chover, J. M. Sotoca L. A. García, “A game engine to make games as multi-agent systems,” *Advances in Engineering Software*, c. 140, s. 102 732, 2020.

- [12] E. Folmer, “Component based game development—a solution to escalating costs and expanding deadlines?” *Component-Based Software Engineering: 10th International Symposium, CBSE 2007, Medford, MA, USA, July 9-11, 2007. Proceedings 10*, Springer, 2007, ss. 66–73.
- [13] P. Zackariasson T. L. Wilson, *The video game industry: Formation, present state, and future*. Routledge, 2012.
- [14] J. Kopf D. Lischinski, “Depixelizing pixel art,” içinde *ACM SIGGRAPH 2011 papers*, 2011, ss. 1–8.
- [15] P. Lankoski, S. Heliö I. Ekman, “Characters in Computer Games: Toward Understanding Interpretation and Design.,” *DiGRA Conference*, c. 2, 2003, ss. 1–12.
- [16] E. Neiva C. Romano, “The semiotic immersion of video games, gaming technology and interactive strategies,” *Public Journal of Semiotics*, c. 1, no. 2, ss. 31–49, 2007.
- [17] E. Grahn, *A study in GUI aesthetics for modern pixel art games*. 2013.
- [18] S.-L. Tian T. Woo, “Study on the Influence of Game Character Design on Users’ Continuous Immersion,” *Journal of Digital Contents Society*, c. 20, no. 6, ss. 1097–1104, 2019.
- [19] J. Cooper, *Game Anim: Video Game Animation Explained*. CRC Press, 2021, isbn: 9781000357806. erişim adresi: <https://books.google.com.tr/books?id=ZMIZEAAAQBAJ>.
- [20] S. Hong, S. Kim S. Kang, “Game Sprite Generator Using a Multi Discriminator GAN.,” *KSII Transactions on Internet & Information Systems*, c. 13, no. 8, 2019.
- [21] J. Togelius, N. Shaker M. J. Nelson, “The search-based approach,” *Computational Synthesis and Creative Systems*, Cham, Springer, ss. 17–30, 2016.
- [22] N. Justesen, P. Bontrager, J. Togelius S. Risi, “Deep learning for video game playing,” *IEEE Transactions on Games*, c. 12, no. 1, ss. 1–20, 2019.
- [23] M. Hendriks, S. Meijer, J. Van Der Velden A. Iosup, “Procedural content generation for games: A survey,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, c. 9, no. 1, ss. 1–22, 2013.
- [24] G. Lai, F. F. Leymarie W. Latham, “On Mixed-Initiative Content Creation for Video Games,” *IEEE Transactions on Games*, c. 14, no. 4, ss. 543–557, 2022.
- [25] A. Summerville ve diğ., “Procedural content generation via machine learning (PCGML),” *IEEE Transactions on Games*, c. 10, no. 3, ss. 257–270, 2018.
- [26] A. Fadaeddini, B. Majidi M. Eshghi, “A case study of generative adversarial networks for procedural synthesis of original textures in video games,” *2018 2nd National and 1st International Digital Games Research Conference: Trends, Technologies, and Applications (DGRC)*, IEEE, 2018, ss. 118–122.

- [27] J. Togelius, N. Shaker M. J. Nelson, “Introduction,” içinde *Procedural Content Generation in Games*. Cham: Springer International Publishing, 2016, ss. 1–15, isbn: 978-3-319-42716-4. doi: 10.1007/978-3-319-42716-4_1. erişim adresi: https://doi.org/10.1007/978-3-319-42716-4%5C_1.
- [28] J. Togelius, N. Shaker J. Dormans, “Grammars and L-systems with applications to vegetation and levels,” içinde *Procedural Content Generation in Games*. Cham: Springer International Publishing, 2016, ss. 73–98, isbn: 978-3-319-42716-4. doi: 10.1007/978-3-319-42716-4_5. erişim adresi: https://doi.org/10.1007/978-3-319-42716-4%5C_5.
- [29] E. Yazan M. F. Talu, “Comparison of the stochastic gradient descent based optimization techniques,” *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, IEEE, 2017, ss. 1–5.
- [30] V. Nagarajan J. Z. Kolter, “Gradient descent GAN optimization is locally stable,” *Advances in Neural Information Processing Systems*, I. Guyon ve diğ., ed., c. 30, Curran Associates, Inc., 2017. erişim adresi: https://proceedings.neurips.cc/paper%5C_files/paper/202017/file/7e0a0209b929d097bd3e8ef30567a5c1-Paper.pdf.
- [31] J. Lu, “Gradient descent, stochastic optimization, and other tales,” *arXiv preprint arXiv:2205.00832*, 2022.
- [32] D. P. Kingma J. Ba, *Adam: A Method for Stochastic Optimization*, 2017. arXiv: 1412.6980 [cs.LG].
- [33] D. Soydaner, “A comparison of optimization algorithms for deep learning,” *International Journal of Pattern Recognition and Artificial Intelligence*, c. 34, no. 13, s. 2052013, 2020.
- [34] I. Goodfellow ve diğ., “Generative adversarial nets,” *Advances in neural information processing systems*, c. 27, 2014.
- [35] A. Gonzalez, M. Guzdial F. Ramos, “Generating gameplay-relevant art assets with transfer learning,” *arXiv preprint arXiv:2010.01681*, 2020.
- [36] I. Goodfellow, Y. Bengio A. Courville, *Deep learning*. MIT press, 2016.
- [37] R. Corvi, D. Cozzolino, G. Zingarini, G. Poggi, K. Nagano L. Verdoliva, “On the detection of synthetic images generated by diffusion models,” *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, ss. 1–5.
- [38] A. Radford, L. Metz S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [39] S. Ioffe C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *International conference on machine learning*, pmlr, 2015, ss. 448–456.
- [40] I. Goodfellow, “Nips 2016 tutorial: Generative adversarial networks,” *arXiv preprint arXiv:1701.00160*, 2016.

- [41] L. Horsley D. Perez-Liebana, “Building an automatic sprite generator with deep convolutional generative adversarial networks,” *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, IEEE, 2017, ss. 134–141.
- [42] M. Mirza S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
- [43] F. C. Silva, P. A. L. de Castro, H. R. Júnior E. C. Marujo, “Mangan: Assisting colorization of manga characters concept art using conditional gan,” *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, ss. 3257–3261.
- [44] H. Ren, J. Li N. Gao, “Two-stage sketch colorization with color parsing,” *IEEE Access*, c. 8, ss. 44 599–44 610, 2019.
- [45] P. Isola, J.-Y. Zhu, T. Zhou A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, ss. 1125–1134.
- [46] O. Ronneberger, P. Fischer T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, Springer, 2015, ss. 234–241.
- [47] U. Demir G. Unal, *Patch-Based Image Inpainting with Generative Adversarial Networks*, 2018. arXiv: 1803.07422 [cs.CV].
- [48] H. Zhao, O. Gallo, I. Frosio J. Kautz, “Loss functions for image restoration with neural networks,” *IEEE Transactions on computational imaging*, c. 3, no. 1, ss. 47–57, 2016.
- [49] Y. R. Serpa M. A. F. Rodrigues, “Towards machine-learning assisted asset generation for games: a study on pixel art sprite sheets,” *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, IEEE, 2019, ss. 182–191.
- [50] B. Xu, N. Wang, T. Chen M. Li, “Empirical evaluation of rectified activations in convolutional network,” *arXiv preprint arXiv:1505.00853*, 2015.
- [51] D.-A. Clevert, T. Unterthiner S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2015.
- [52] Z. Jiang P. Sweetser, “GAN-Assisted YUV Pixel Art Generation,” *Australasian Joint Conference on Artificial Intelligence*, Springer, 2022, ss. 595–606.
- [53] G. Wang, R. Fu, B. Sun, J. Lv, T. Sheng Y. Tan, “Comparison of two types of color transfer algorithms in YUV and Lab color spaces,” *AOPC 2017: Optical Sensing and Imaging Technology and Applications*, SPIE, c. 10462, 2017, ss. 746–752.
- [54] H. Zeng, X. Zhang, Z. Yu Y. Wang, “SR-ITM-GAN: Learning 4K UHD HDR with a generative adversarial network,” *IEEE Access*, c. 8, ss. 182 815–182 827, 2020.

- [55] F. Coutinho L. Chaimowicz, “Generating Pixel Art Character Sprites using GANs,” *2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 2022, ss. 1–6. doi: 10 . 1109 / SBGAMES56371.2022.9961120.
- [56] S. Ioffe, “Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models,” *Advances in Neural Information Processing Systems*, I. Guyon ve diğ., ed., c. 30, Curran Associates, Inc., 2017. erişim adresi: https://proceedings.neurips.cc/paper/5C_files/paper/202017/file/c54e7837e0cd0ced286cb5995327d1ab-Paper.pdf.
- [57] C. W. Seo Y. Seo, “Seg2pix: Few shot training line art colorization with segmented image data,” *Applied Sciences*, c. 11, no. 4, s. 1464, 2021.
- [58] S.-H. Zhang, T. Chen, Y.-F. Zhang, S.-M. Hu R. R. Martin, “Vectorizing cartoon animations,” *IEEE Transactions on Visualization and Computer Graphics*, c. 15, no. 4, ss. 618–629, 2009.
- [59] H. Zhang, I. Goodfellow, D. Metaxas A. Odena, “Self-attention generative adversarial networks,” *International conference on machine learning*, PMLR, 2019, ss. 7354–7363.
- [60] S. Targ, D. Almeida K. Lyman, “Resnet in resnet: Generalizing residual architectures,” *arXiv preprint arXiv:1603.08029*, 2016.
- [61] A. Elgammal, B. Liu, M. Elhoseiny M. Mazzone, “Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms,” *arXiv preprint arXiv:1706.07068*, 2017.
- [62] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen T. Aila, “Analyzing and improving the image quality of stylegan,” *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, ss. 8110–8119.
- [63] T. Karras, S. Laine T. Aila, “A style-based generator architecture for generative adversarial networks,” *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, ss. 4401–4410.
- [64] X. Huang S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” *Proceedings of the IEEE international conference on computer vision*, 2017, ss. 1501–1510.
- [65] K. Schutte, O. Moindrot, P. Hérent, J.-B. Schiratti S. Jégou, “Using stylegan for visual interpretability of deep learning models on medical images,” *arXiv preprint arXiv:2101.07563*, 2021.
- [66] R. Abdal, P. Zhu, N. J. Mitra P. Wonka, “Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows,” *ACM Transactions on Graphics (ToG)*, c. 40, no. 3, ss. 1–21, 2021.
- [67] Y. Lu, S. Wu, Y.-W. Tai C.-K. Tang, “Image generation from sketch constraint using contextual gan,” *Proceedings of the European conference on computer vision (ECCV)*, 2018, ss. 205–220.
- [68] S. Kullback R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, c. 22, no. 1, ss. 79–86, 1951.

- [69] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang A. A. Efros, “Learning dense correspondence via 3d-guided cycle consistency,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, ss. 117–126.
- [70] D. Ulyanov, A. Vedaldi V. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” *arXiv preprint arXiv:1607.08022*, 2016.
- [71] *pokemon2pokemon: Using Neural Networks to Generate Pokemon as Different Elemental Types*. erişim adresi: <https://www.rileynwong.com/blog/2019/5/22/%20pokemon2pokemon-using-cyclegan-to-generate-pokemon-as%20%5C%5C-different-elemental-types> (erişim tarihi 05/09/2024).
- [72] T. Kim, M. Cha, H. Kim, J. K. Lee J. Kim, “Learning to discover cross-domain relations with generative adversarial networks,” *International conference on machine learning*, PMLR, 2017, ss. 1857–1865.
- [73] E. Şahin M. F. Talu, “Wy-Net: A New Approach To Image Synthesis With Generative Adversarial Networks,” *Journal of Scientific Reports-A*, no. 050, ss. 270–290, 2022.
- [74] T. Angsarawanee B. Kijsirikul, “Generating images with desired properties using the DiscoGAN model enhanced with repeated property construction,” *Proceedings of the 1st International Conference on Advanced Information Science and System*, 2019, ss. 1–9.
- [75] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim J. Choo, “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, ss. 8789–8797.
- [76] H. Lee, A. Battle, R. Raina A. Ng, “Efficient sparse coding algorithms,” *Advances in neural information processing systems*, c. 19, 2006.
- [77] B. A. Olshausen D. J. Field, “Emergence of simple-cell receptive field properties by learning a sparse code for natural images,” *Nature*, c. 381, no. 6583, ss. 607–609, 1996.
- [78] C. Doersch, “Tutorial on variational autoencoders,” *arXiv preprint arXiv:1606.05908*, 2016.
- [79] D. P. Kingma M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [80] A. v. d. Oord, O. Vinyals K. Kavukcuoglu, “Neural discrete representation learning,” *arXiv preprint arXiv:1711.00937*, 2017.
- [81] Z. Chen ve diğ., “Cancer subtyping by improved transcriptomic features using vector quantized variational autoencoder,” *arXiv preprint arXiv:2207.09783*, 2022.
- [82] W. Yan, Y. Zhang, P. Abbeel A. Srinivas, *VideoGPT: Video Generation using VQ-VAE and Transformers*, 2021. arXiv: 2104.10157 [cs.CV].
- [83] A. Saravanan M. Guzdial, “Pixel vq-vaes for improved pixel art representation,” *arXiv preprint arXiv:2203.12130*, 2022.

- [84] B. Ivanovic, K. Leung, E. Schmerling M. Pavone, “Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach,” *IEEE Robotics and Automation Letters*, c. 6, no. 2, ss. 295–302, 2020.
- [85] C. Zhang, R. Barbano B. Jin, “Conditional variational autoencoder for learned image reconstruction,” *Computation*, c. 9, no. 11, s. 114, 2021.
- [86] K. Sohn, H. Lee X. Yan, “Learning Structured Output Representation using Deep Conditional Generative Models,” *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama R. Garnett, ed., c. 28, Curran Associates, Inc., 2015. erişim adresi: https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.
- [87] C. Furusawa, K. Hiroshiba, K. Ogaki Y. Odagiri, “Comicolorization: semi-automatic manga colorization,” içinde *SIGGRAPH Asia 2017 Technical Briefs*, 2017, ss. 1–4.
- [88] S. Roy S. K. Bandyopadhyay, “Face detection using a hybrid approach that combines HSV and RGB,” *International Journal of Computer Science and Mobile Computing*, c. 2, no. 3, ss. 127–136, 2013.
- [89] *TinyHero - Retro pixel characters dataset*. erişim adresi: <https://www.kaggle.com/datasets/calmness/retro-pixel%20-characters-generator> (erişim tarihi 05/09/2024).
- [90] N. Inkawhich, *DCGAN Tutorial*, 2018. erişim adresi: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html#dcgan-tutorial (erişim tarihi 13/03/2025).
- [91] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford X. Chen, “Improved techniques for training gans,” *Advances in neural information processing systems*, c. 29, 2016.
- [92] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” *Advances in neural information processing systems*, c. 30, 2017.

A

Karakter Renklendirme Deneyinin Detayları

A.1 Değerlendirme

Karakter renklendirme deneyimizde Bölüm 3.1.2’de mimarilerin sonuçlarını OKH yöntemiyle değerlendirdik. OKH yönteminin bize göre kötü sonuçlar verdiği durumlarda, fikirlerimizi belirttik. Mimarileri ürettikleri görseller üzerinden OKH yöntemi ve tek tek inceleyerek değerlendirmemizin temel nedeni, FBM yöntemi ile değerlendirme yapmanın karakter üretiminde bahsettiğimiz (Bölüm 3.2.3) nedenlerden dolayı yetersiz kalmasıdır. Literatürde yapılan çalışmalarda da üretilen görüntüler detaylı bir şekilde incelenmiş ve yapılan katkılar görüntüler üzerinden anlatılmaya çalışılmıştır.

Üretilen görsellere ait şeffaflık oluşturma için Bölüm 3.1.2’de yaptığımız mimarilere ait karşılaştırmaların FBM skorlarını Tablo A.1’de paylaştık.

Tablo A.1 Bölüm 3.1.2’de kullandığımız mimarilerin ürettiği çıktıları ait FBM skorları. 20.000 - 40.000 adım aralığındaki çıktıları kullandık.

Mimari	FBM Skoru
Isola vd. [45] Pix2Pix Mimarisi	69,04
Serpa ve Rodrigues [49] Pix2Pix Mimarisi	121,64
Jiang ve Sweetser [52] Pix2Pix Mimarisi	80,3
Coutinho ve Chaimowicz [55] Pix2Pix Mimarisi	82,31

Tablo A.1’de paylaştığımız FBM skorlarına göre Isola vd. [45] Pix2Pix mimarisi en kaliteli ve orijinal görüntülere en yakın çıktıları üretmektedir.

Isola vd. [45] tarafından ortaya konan orijinal Pix2Pix mimarisi üzerine yaptığımız iyileştirmelerimiz sonucu ortaya koyduğumuz mimarimize ait FBM skor değerini Tablo A.2’de paylaştık. FBM değerlendirme yöntemine göre daha kötü bir skor elde ettiğimiz açıkça görülse de minimal değişimlerin önünü açtık. Aynı zamanda daha iyi çıktı almak için daha uzun eğitim sürelerine ihtiyacımız olduğundan bahsetmiştik. Mimarimizin daha uzun eğitim süresine ihtiyaç duyması da aldığımız kötü sonuca etki etmiştir.

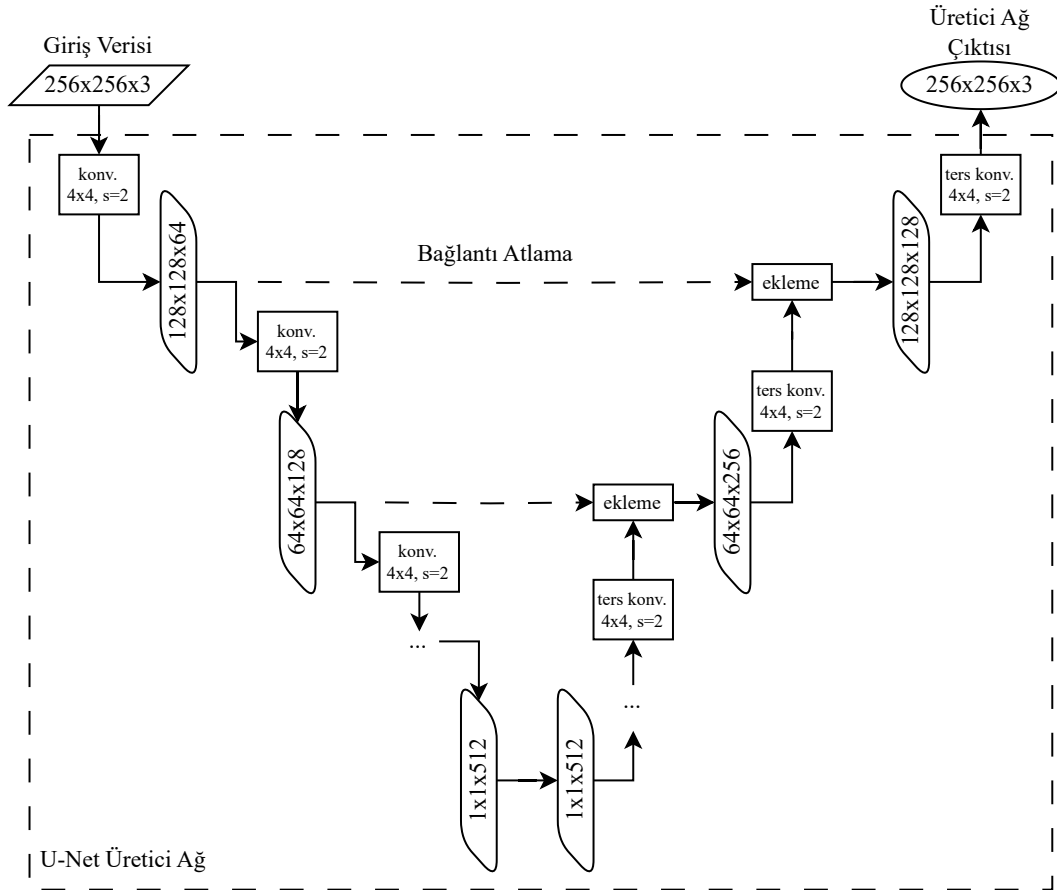
Tablo A.2 Mimarimize ait FBM skoru. 20.000 - 40.000 adım aralığındaki çıktıları kullanıyoruz.

Mimari	FBM Skoru
Isola vd. [45] Pix2Pix Mimarisi	69,04
Ortaya Koyduğumuz Line2Pix Mimarisi	76,4

A.2 Mimari

Bölüm 3.1.4'te ortaya koyduğumuz mimarinin detaylarını paylaştık. Anlaşılabilirliği artırmak amacıyla yararlandığımız orijinal Pix2Pix mimarisine ait detayları da paylaştık. Eklèmelerimizi kırmızı renkle renklendirerek daha anlaşılır biçimde göstermeye çalıştık.

A.2.1 Üretici Ağ

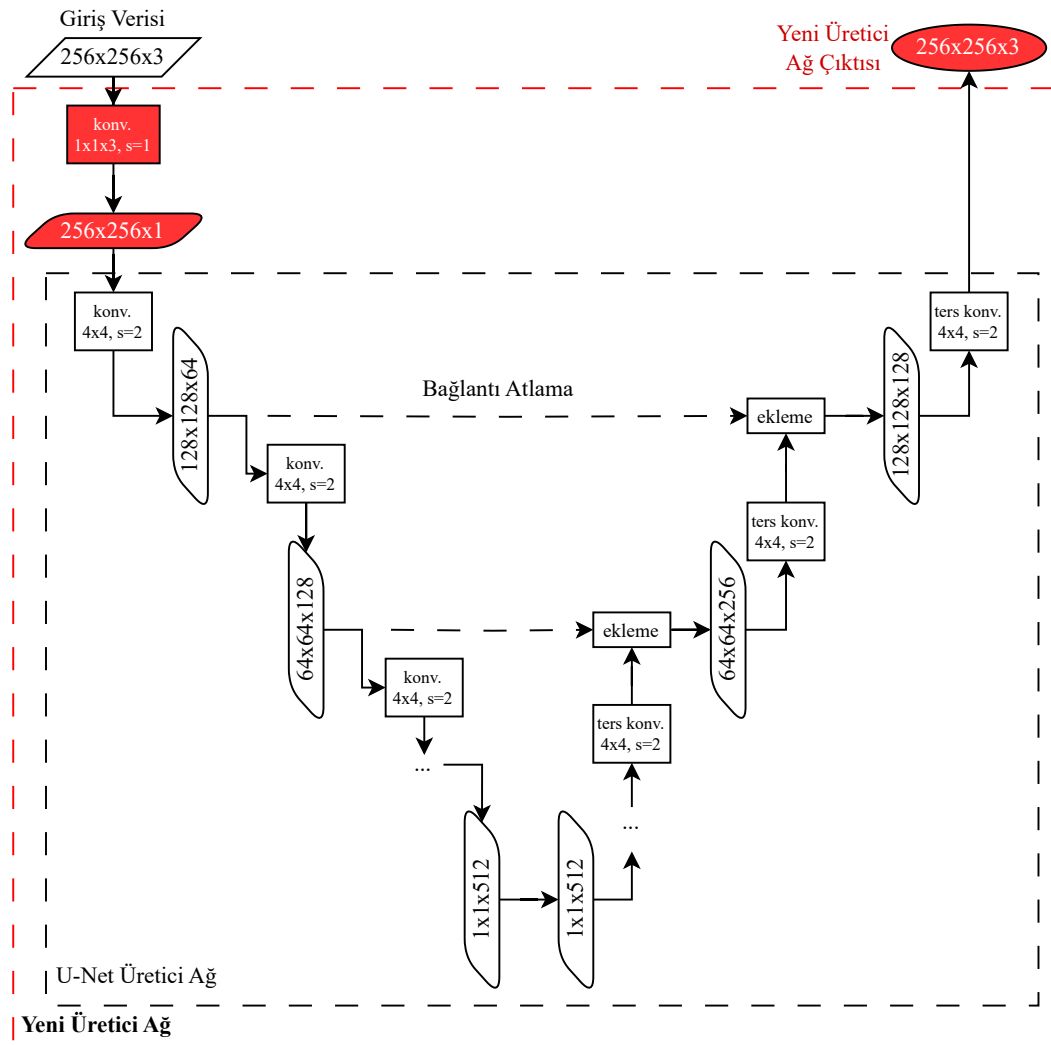


Şekil A.1 Isola vd. [45] Pix2Pix mimarisinde kullanılan üretici ağ.

Isola vd. [45] tarafından ortaya konan Pix2Pix mimarisinde kullanılan üretici ağ mimarisini Şekil A.1'de paylaştık.

Isola vd. [45] tarafından ortaya konan Pix2Pix mimarisinde iyileştirmeler yaparak ortaya koyduğumuz Pix2Pix mimarisinde kullandığımız üretici ağ mimarisini Şekil

A.2’de paylaştık.



Şekil A.2 Ortaya koyduğumuz Line2Pix mimarisinde kullanılan üretici ağ. Eklèmelerimiz kırmızı renkle renklendirilmiştir.

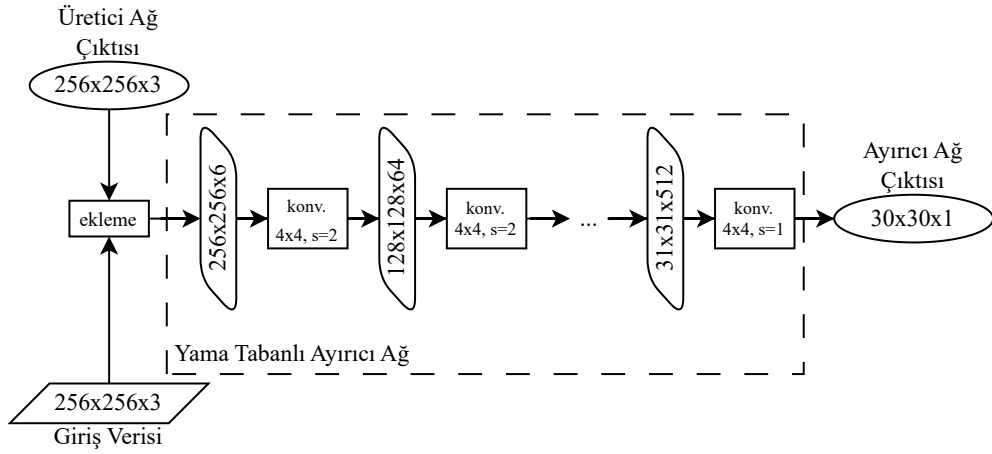
A.2.2 Ayırıcı Ağ

Isola vd. [45] tarafından ortaya konan Pix2Pix mimarisinde kullanılan ayırıcı ağ mimarisini Şekil A.3'te paylaştık.

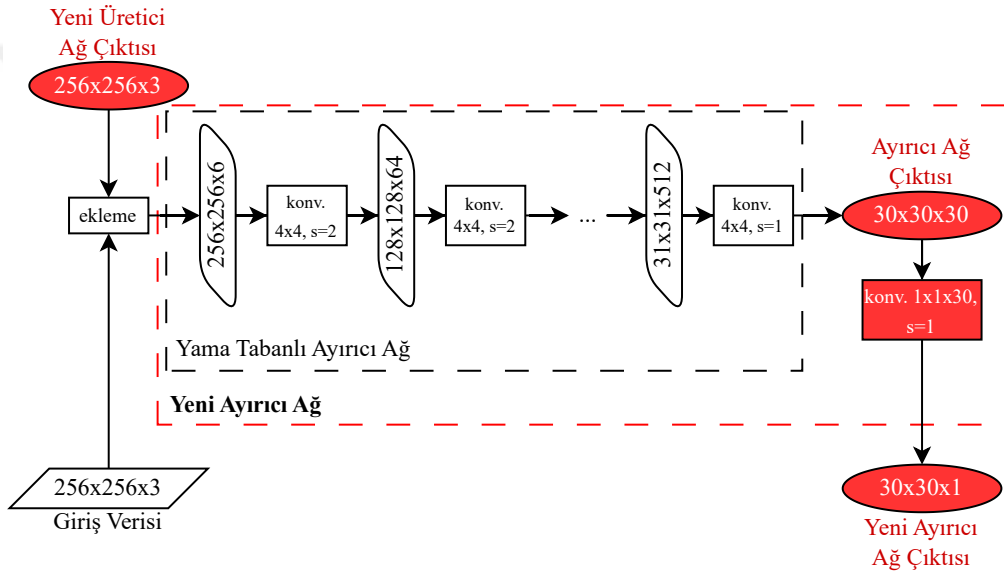
Isola vd. [45] tarafından ortaya konan Pix2Pix mimarisinde iyileştirmeler yaparak ortaya koyduğumuz Pix2Pix mimarisinde kullandığımız ayırıcı ağ mimarisini Şekil A.4'te paylaştık.

A.3 Mimarilere Ait Kodlama Detayları

Bölüm 3.1.4'te ortaya koyduğumuz mimarinin detaylarını paylaştık. Daha detaylı bir açıklama gösterebilmek adına bazı kod detaylarını sunduk.



Şekil A.3 Isola vd. [45] Pix2Pix mimarisinde kullanılan ayırıcı ağ.



Şekil A.4 Ortaya koyduğumuz Line2Pix mimarisinde kullanılan ayırıcı ağ. Eklemelerimiz kırmızı renkle renklendirilmiştir.

Üretici ağ içerisinde kullandığımız kod bloğunu Sözde Kod A.1’de detaylandırıyoruz. Kalın harflerle yazılan kod satırları üretici ağ mimarisine eklediğimiz değişiklikleri temsil etmektedir.

Ayırıcı ağ içerisinde kullandığımız kod bloğunu Sözde Kod A.2’de detaylandırıyoruz. Kalın harflerle yazılan kod satırları ayırıcı ağ mimarisine eklediğimiz değişiklikleri temsil etmektedir.

```

1 üretici ağı başlat:
2     girdi katmanı boyutu: [256, 256, 3]
3     alt örnekleme katmanları oluştur:
4         alt örnekleme (1, 1, 1, Leaky ReLU, yok)
5         alt örnekleme (64, 4, 2, Leaky ReLU, var)
6         alt örnekleme (128, 4, 2, Leaky ReLU, var)
7         alt örnekleme (256, 4, 2, Leaky ReLU, var)
8         alt örnekleme (512, 4, 2, Leaky ReLU, var)
9         alt örnekleme (512, 4, 2, Leaky ReLU, var)
10        alt örnekleme (512, 4, 2, Leaky ReLU, var)
11        alt örnekleme (512, 4, 2, Leaky ReLU, var)
12        alt örnekleme (512, 4, 2, Leaky ReLU, var)
13    üst örnekleme katmanları oluştur:
14        üst örnekleme (512, 4, 2, ReLU, var, 0.5)
15        üst örnekleme (512, 4, 2, ReLU, var, 0.5)
16        üst örnekleme (512, 4, 2, ReLU, var, 0.5)
17        üst örnekleme (512, 4, 2, ReLU, var, yok)
18        üst örnekleme (256, 4, 2, ReLU, var, yok)
19        üst örnekleme (128, 4, 2, ReLU, var, yok)
20        üst örnekleme (64, 4, 2, ReLU, var, yok)
21        üst örnekleme (3 (çıktı kanalı), 4, 2, ReLU, yok, yok)

```

Sözde Kod A.1 Üretici ağ içerisindeki kod bloğuna ait sözde kod. Alt örnekleme katmanları için: alt örnekleme (filtre, çekirdek boyutu, adım, aktivasyon fonksiyonu, toplu normalleştirme). Üst örnekleme katmanları için: (filtre, çekirdek boyutu, adım, aktivasyon fonksiyonu, toplu normalleştirme, bırakma (dropout) değeri).

```

1 ayırıcı ağı başlat:
2     girdi katmanı boyutu: [256, 256, 3]
3     hedef (koşul) katmanı boyutu: [256, 256, 3]
4     ayırıcı ağ girdi (girdi + hedef) boyutu: [256, 256, 6]
5     alt örnekleme (64, 4, 2, Leaky ReLU, yok)
6     alt örnekleme (128, 4, 2, Leaky ReLU, var)
7     alt örnekleme (256, 4, 2, Leaky ReLU, var)
8     alt örnekleme (512, 4, 1, Leaky ReLU, var)
9     alt örnekleme (30, 4, 1, Leaky ReLU, var)
10    alt örnekleme (1, 1, 1, yok, yok)

```

Sözde Kod A.2 Ayırıcı ağ içerisindeki kod bloğuna ait sözde kod. Alt örnekleme katmanları için: alt örnekleme (filtre, çekirdek boyutu, adım, aktivasyon fonksiyonu, toplu normalleştirme). Boşluk için: boşluk (yükseklik, genişlik).

B

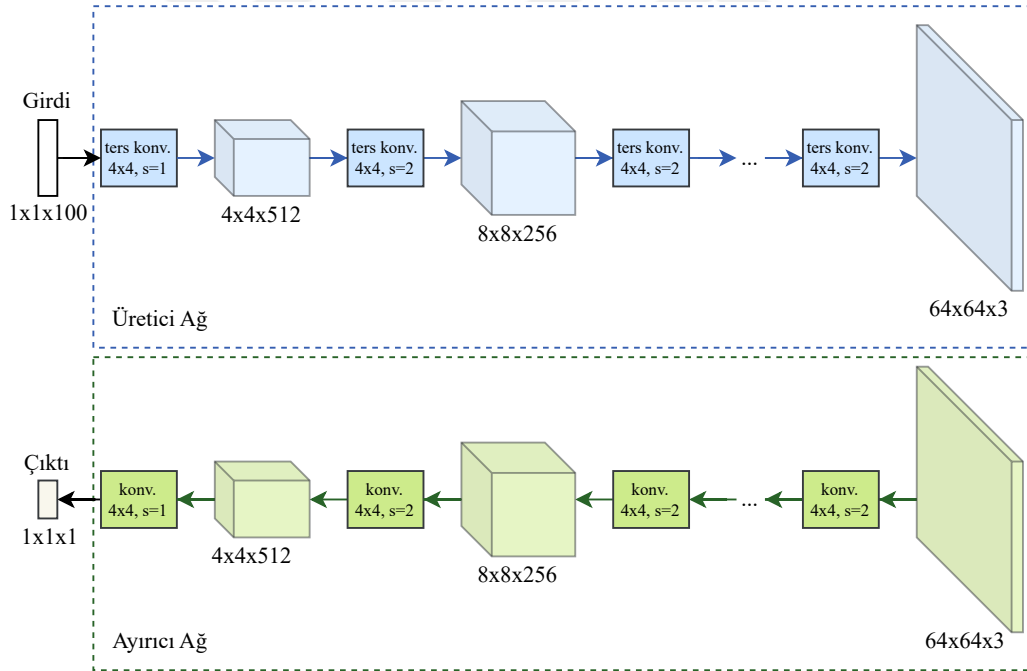
Karakter Üretimi Deneyinin Detayları

B.1 Mimari Detayları

B.1.1 Mimarilerin Görselleştirilmesi

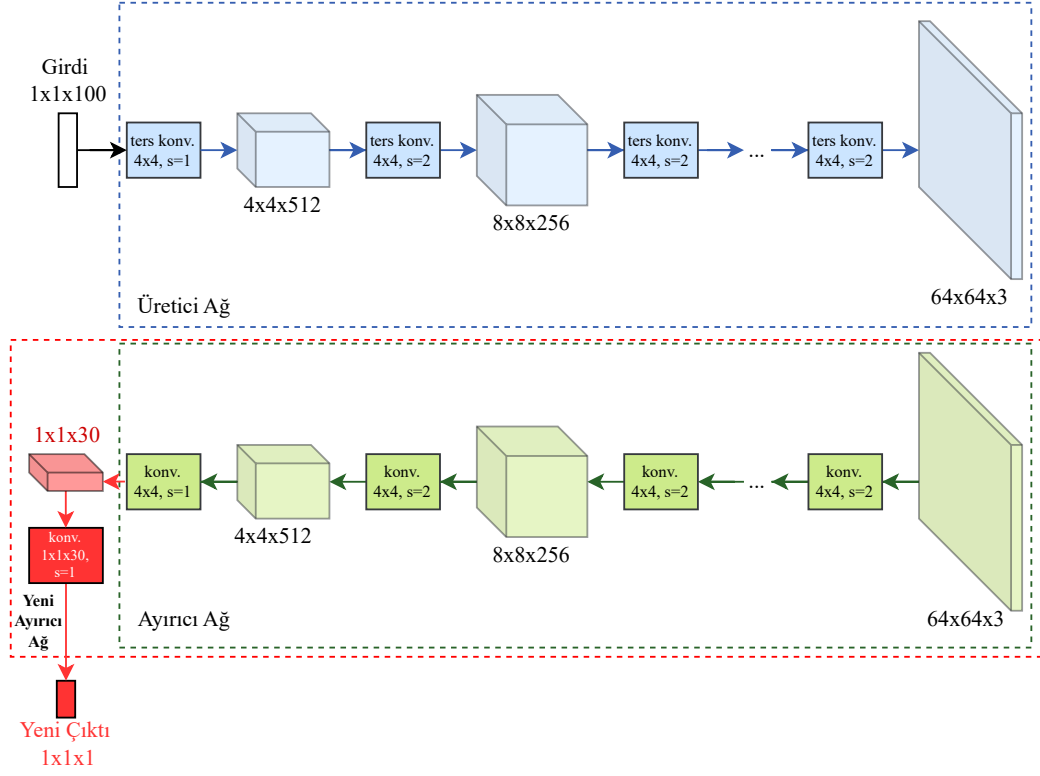
DKÇÜA mimarisi [38, 90] ile DKÇÜA mimarisinde iyileştirmeler yaparak ortaya koyduğumuz DKÇÜA mimarisine ait görseller sunduk. Mimarilerin ve yaptığımız çalışmanın daha iyi anlaşılmasını sağlamaya çalıştık.

Kullandığımız DKÇÜA mimarisini [90] detaylı olarak Şekil B.1’de paylaştık.



Şekil B.1 Kullandığımız DKÇÜA mimarisi [90].

DKÇÜA mimarisine yaptığımız iyileştirmelerle ortaya koyduğumuz DKÇÜA mimarisini detaylı olarak Şekil B.2’de paylaştık.



Şekil B.2 Ortaya koyduğumuz DKÇÜA mimarisi. Eklmelerimiz kırmızı renkle renklendirilmiştir.

B.1.2 Mimarilere Ait Kodlamalar

Bölüm 3.2’de ele aldığımız karakter üretimi deneyinde oluşturduğumuz mimarinin kodlamalarını paylaştık. Mimariye ait üretici ağ sözde kod bloğunu Sözde Kod B.1’de gösterdik.

```

1 üretici ağı başlat:
2   girdi katmanı boyutu: [1, 1, 100]
3   üst örnekleme (100, 512, 4, 1, 0, ReLU, var)
4   üst örnekleme (512, 256, 4, 2, 1, ReLU, var)
5   üst örnekleme (256, 128, 4, 2, 1, ReLU, var)
6   üst örnekleme (128, 64, 4, 2, 1, ReLU, var)
7   üst örnekleme (64, 3, 4, 2, 1, tanh, yok)

```

Sözde Kod B.1 Üretici ağ içerisindeki kod bloğuna ait sözde kod. Üst örnekleme katmanları için: üst örnekleme (girdi boyutu, çıktı boyutu, çekirdek boyutu, adım, boşluk, aktivasyon fonksiyonu, toplu normalleştirme).

Mimariye ait ayırıcı ağ sözde kod bloğunu Sözde Kod B.2’de gösterdik. Kalın harflerle yazılan kod satırları ayırıcı ağ mimarisine eklediğimiz değişiklikleri temsil etmektedir.

```
1 ayırıcı ağı başlat:
2   girdi katmanı boyutu: [64, 64, 3]
3   alt örnekleme (3, 64, 4, 2, 1, Leaky ReLU, yok)
4   alt örnekleme (64, 128, 4, 2, 1, Leaky ReLU, var)
5   alt örnekleme (128, 256, 4, 2, 1, Leaky ReLU, var)
6   alt örnekleme (256, 512, 4, 2, 1, Leaky ReLU, var)
7   alt örnekleme (512, 30, 4, 1, 0, Leaky ReLU, yok)
8   alt örnekleme (30, 1, 1, 1, 0, sigmoid, yok)
```

Sözde Kod B.2 Ayırıcı ağ içerisindeki kod bloğuna ait sözde kod. Alt örnekleme katmanları için: alt örnekleme (girdi boyutu, çıktı boyutu, çekirdek boyutu, adım, boşluk, aktivasyon fonksiyonu, toplu normalleştirme).

B.2 Ankete İlişkin Detaylar

Bu bölümde hazırladığımız anketin detaylarını paylaştık. Ankete ait soruları ve kullanıcıya yönelttiğimiz bu soruların cevaplarına ait detaylarını paylaşarak anket hakkında şeffaf olduğumuzu göstermeyi amaçladık.

Anketi hazırlarken seçtiğimiz karakterlerin, mimariler tarafından üretilen en kaliteli çıktılar arasından seçildiği unutulmamalıdır. Kaliteli çıktılar arasında da birbirine benzer olan çıktıları kullanmamayı tercih ettik. Her bir mimari için 16 farklı çıktı seçtik ve seçtiğimiz çıktılar ile 5 farklı form oluşturduk. Anket sonuçlarının bir bütün olarak değerlendirildiği tabloya Bölüm 3.2.3'te (Tablo 3.7) değinilmiştir.

B.2.1 1. Form

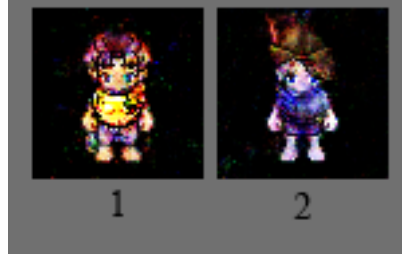
İlk formumuzu detaylı olarak açıklamanın anlaşılabilirlik açısından daha iyi olacağını düşünüyoruz. Hazırladığımız diğer 5 form, 1. form üzerinde yaptığımız işlemlere benzer işlemlere sahiptir. 1. formda kullanılan mimarilerden elde edilmiş karakterleri Şekil B.3'te inceleyebilmeniz için sunduk. Şekil B.3 üzerinde bulunan her bir görüntü (Şekil B.3.1) yöneltilen soru grubuna ait mimarilerin oluşturduğu karakterleri göstermektedir.

Şekil B.3 ile formda kullanılan her soru grubuna ait olan mimarilerin çıktılarından elde edilen görüntüleri paylaştık. Kullanıcılar anketteki sorularda görüntü üzerindeki numaralandırılmış karakterlerden birini tercih etmiştir. Her formda olduğu gibi 1. formda da 5 farklı görüntü ile beraber soru gruplarını oluşturduk. Kullanılan görüntülerdeki karakterlerin sıralamasını daha güvenilir bir anket oluşturmak amacıyla değiştirdik. Değişiklik sonucu Şekil B.3'te bulunan görüntülerdeki karakterlerin hangi mimariyle ait olduğunu Tablo B.1'de gösterdik.

1. form ile anketimize 13 kullanıcı katılmıştır. Kullanıcıların 1. form görüntülerine karşılık verdiği oyların sayılarını ve yüzdelerini Tablo B.2'de paylaştık.



(1)



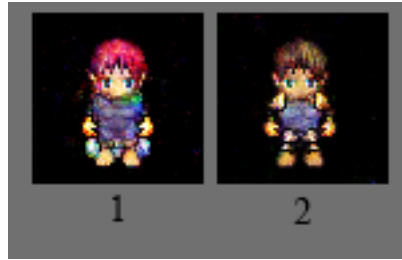
(2)



(3)



(4)



(5)

Şekil B.3 1. formda kullanılan karakter grupları.

Tablo B.1 Şekil B.3'te bulunan karakterlerin ait olduğu mimariler.

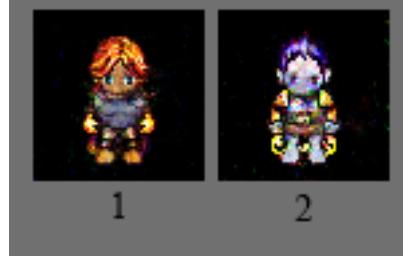
Karakter Grubu	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Şekil B.3.1	1	2
Şekil B.3.2	2	1
Şekil B.3.3	2	1
Şekil B.3.4	2	1
Şekil B.3.5	1	2

Tablo B.2 1. forma ait sonuçlar.

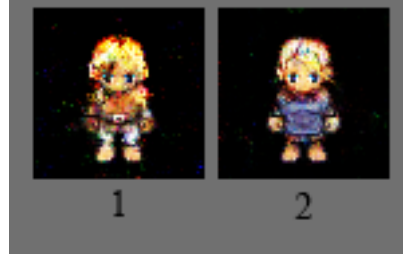
Değerlendirme Alanı	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Görüntü Kalitesi	35 (%54)	30 (%46)
Ana Hat Belirginliği	32 (%49)	33 (%51)
Renk Uyumu	37 (%57)	28 (%43)
Tercih Edilen Karakter	32 (%49)	33 (%51)

B.2.2 2. Form

2. formumuza 11 kişi katılım sağlamıştır. Şekil B.4'te formda kullanılan karakter gruplarını gösterdik. Karakterlerin hangi mimarilere ait olduğuna dair bilgiyi Tablo B.3'te paylaştık. 2. forma katılan kullanıcıların verdiği oylarla ortaya çıkan sonuçları Tablo B.4'te sunduk.



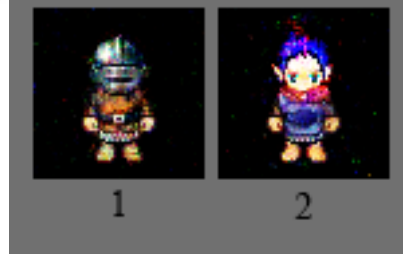
(1)



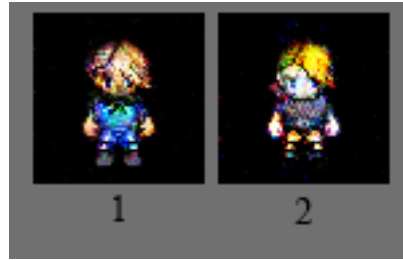
(2)



(3)



(4)



(5)

Şekil B.4 2. formda kullanılan karakter grupları.

Tablo B.3 Şekil B.4’te bulunan karakterlerin ait olduğu mimariler.

Karakter Grubu	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Şekil B.4.1	2	1
Şekil B.4.2	2	1
Şekil B.4.3	2	1
Şekil B.4.4	1	2
Şekil B.4.5	1	2

Tablo B.4 2. forma ait sonuçlar.

Değerlendirme Alanı	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Görüntü Kalitesi	20 (%36)	35 (%64)
Ana Hat Belirginliği	20 (%36)	35 (%64)
Renk Uyum	21 (%38)	34 (%62)
Tercih Edilen Karakter	21 (%38)	34 (%62)

B.2.3 3. Form

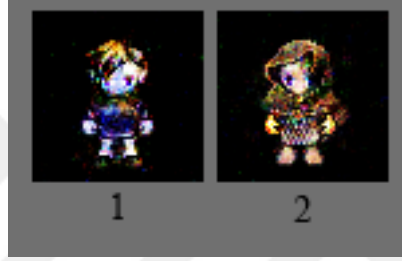
3. formumuza 14 kişi katılım sağlamıştır. Şekil B.5’te formda kullanılan karakter gruplarını gösterdik. Karakterlerin hangi mimarilere ait olduğuna dair bilgiyi Tablo B.5’te paylaştık. 3. forma katılan kullanıcıların verdiği oylarla ortaya çıkan sonuçları Tablo B.6’da sunduk.



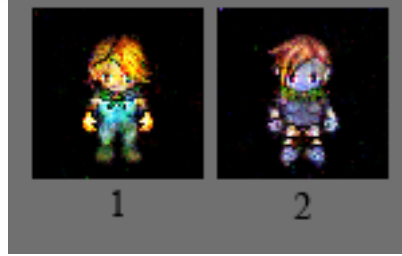
(1)



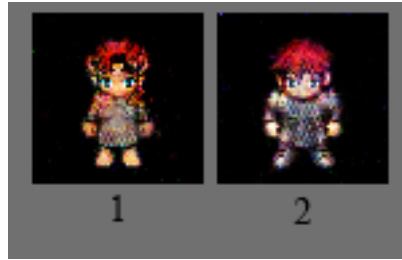
(2)



(3)



(4)



(5)

Şekil B.5 3. formda kullanılan karakter grupları.

Tablo B.5 Şekil B.5’te bulunan karakterlerin ait olduğu mimariler.

Karakter Grubu	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Şekil B.5.1	2	1
Şekil B.5.2	1	2
Şekil B.5.3	1	2
Şekil B.5.4	2	1
Şekil B.5.5	2	1

Tablo B.6 3. forma ait sonuçlar.

Değerlendirme Alanı	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Görüntü Kalitesi	31 (%44)	39 (%56)
Ana Hat Belirginliği	32 (%46)	38 (%54)
Renk Uyumunu	27 (%39)	43 (%61)
Tercih Edilen Karakter	24 (%34)	46 (%66)

B.2.4 4. Form

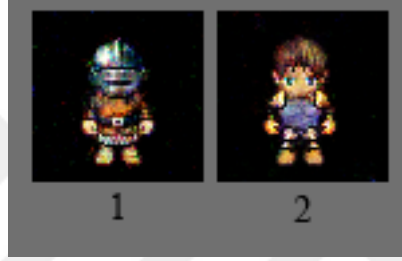
4. formumuza 7 kişi katılım sağlamıştır. Şekil B.5’te formda kullanılan karakter gruplarını gösterdik. Karakterlerin hangi mimarilere ait olduğuna dair bilgiyi Tablo B.7’de paylaştık. 4. forma katılan kullanıcıların verdiği oylarla ortaya çıkan sonuçları Tablo B.8’de sunduk.



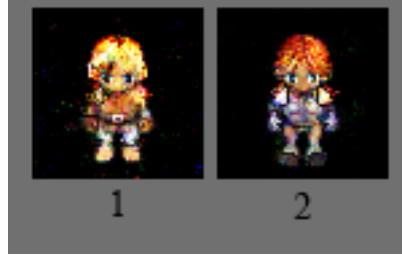
(1)



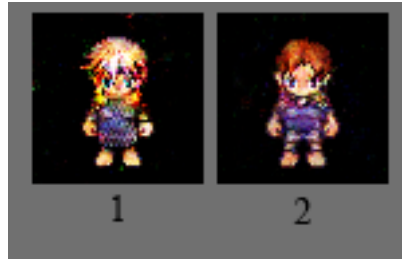
(2)



(3)



(4)



(5)

Şekil B.6 4. formda kullanılan karakter grupları.

Tablo B.7 Şekil B.6’da bulunan karakterlerin ait olduğu mimariler.

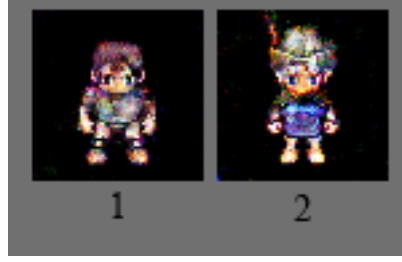
Karakter Grubu	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Şekil B.6.1	2	1
Şekil B.6.2	1	2
Şekil B.6.3	1	2
Şekil B.6.4	2	1
Şekil B.6.5	1	2

Tablo B.8 4. forma ait sonuçlar.

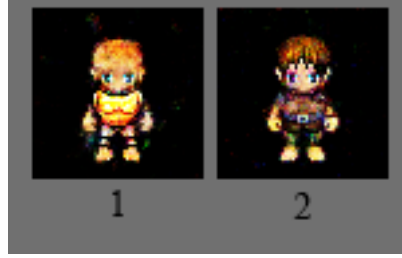
Değerlendirme Alanı	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Görüntü Kalitesi	13 (%37)	22 (%63)
Ana Hat Belirginliği	14 (%40)	21 (%60)
Renk Uyumu	12 (%34)	23 (%66)
Tercih Edilen Karakter	14 (%40)	21 (%60)

B.2.5 5. Form

5. formumuza 6 kişi katılım sağlamıştır. Şekil B.7’de formda kullanılan karakter gruplarını gösterdik. Karakterlerin hangi mimarilere ait olduğuna dair bilgiyi Tablo B.9’da paylaştık. 5. forma katılan kullanıcıların verdiği oylarla ortaya çıkan sonuçları Tablo B.10’da sunduk.



(1)



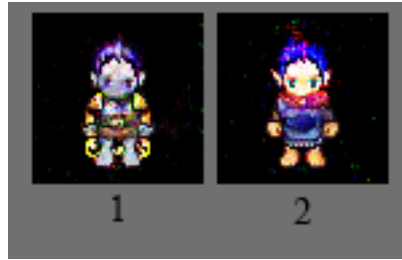
(2)



(3)



(4)



(5)

Şekil B.7 5. formda kullanılan karakter grupları.

Tablo B.9 Şekil B.7’de bulunan karakterlerin ait olduğu mimariler.

Karakter Grubu	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Şekil B.7.1	1	2
Şekil B.7.2	1	2
Şekil B.7.3	1	2
Şekil B.7.4	2	1
Şekil B.7.5	1	2

Tablo B.10 5. forma ait sonuçlar.

Değerlendirme Alanı	DKÇÜA	İyileştirmelerimiz Sonucu DKÇÜA
Görüntü Kalitesi	10 (%33)	20 (%67)
Ana Hat Belirginliği	10 (%33)	20 (%67)
Renk Uyumu	11 (%37)	19 (%63)
Tercih Edilen Karakter	9 (%30)	21 (%70)

2 dimensional (2D)
2 boyutlu (2B)

3 dimensional (3D)
3 boyutlu (3B)

adaptive instance normalization (AdaIN)
uyarlanabilir örnek normalizasyon

adaptive threshold method
uyarlanabilir eşik yöntemi

adversarial loss
çekişme kaybı

affine transformation
afin dönüşüm

anime face dataset (AFC)
anime yüz veri seti (AYV)

art style
sanat stili

asset
sanat varlığı

attention map
dikkat haritası

autoencoder
otomatik kodlayıcı

batch normalization
toplu normalleştirme

batch size
yığın boyutu

blur bulanıklık

byte bit

codebook
kod defteri

codebook loss
kod defteri kaybı

channel
kanal

chrominance blue
mavi renk bileşeni

chrominance red
kırmızı renk bileşeni

classification loss
sınıflama kaybı

color hint
renk ipucu

color space
renk uzayı

commitment loss
bağlılık kaybı

condition
koşul

conditional generative adversarial network (CGAN)
koşullu Çekişmeli Üretici Ağ (KÇÜA)

contextual generative adversarial network
bağlamsal çekişmeli üretici ağ (BÇÜA)

contextual loss
bağlamsal kayıp

convolutions
evrişim, konvolüsyon

creative generative adversarial network (CAN)
yaratıcı çekişmeli üretici ağ (YÇÜA)

cycle-consistency loss
döngü tutarlılık kaybı

cycle-consistent generative adversarial network (CycleGAN)
döngü tutarlı çekişmeli üretici ağ

decoder
çözücü

deep convolutional neural network (DCGAN)

derin konvolüsyonel çekişmeli üretici ağ (DKÇÜA)

discover cross-domain relations with generative adversarial networks (DiscoGAN)

alanlar arası ilişkileri keşfeden çekişmeli üretici ağ

discriminator

ayırıcı ağ

downsampling

alt örnekleme

dropout

bırakma

edge detection

kenar algılama

embedded space

gömülü uzay

encoder

kodlayıcı

epoch

döngü

filter

filtre

Fréchet inception distance (FID)

Fréchet başlangıç mesafesi (FBM)

game engine

oyun motoru

game theory

oyun teorisi

generative adversarial network (GAN)

çekişmeli üretici ağ (ÇÜA)

generator

üretici ağ

Hadamard product

Hadamard çarpımı

historical averaging method

tarihsel ortalama yöntemi

hue (H)

ton

hyperparameter

hiper parametre

image to image (I2I)

görüntüden görüntüye

inception score (IS)

başlangıç skoru (BS)

index

indeks

input

girdi

instance normalization

örnek normalizasyon

kernel size

çekirdek boyutu

key matrix

anahtar matrisi

Kullback-Leibler divergence

Kullback-Leibler sapması

label

etiket

latent space

gizli uzay

latent variable

gizli değişken

learning rate

öğrenme katsayısı

line art

çizgi sanatı

loss function

kayıp fonksiyonu

luminance

parlaklık

mean absolute error (MAE)

ortalama mutlak hata (OMH)

mean squared error (MSE)

ortalama karesel hata (OKH)

minibatch

küçük yığın

modulation

modülasyon

multi discriminator generative adversarial network (MDGAN)

çok ayırıcılı çekişmeli üretici ağ (ÇAÇÜA)

non-player character (NPC)

oyuncu olmayan karakter

normalization

normalizasyon

objective function

maliyet fonksiyonu

one-hot-encoded

tekil kodlama

output

çıktı

patch based

yama tabanlı

perceptual loss

algısal kayıp

pixel

piksel

pixel art

piksel sanatı

pixel-level parsing

piksel düzeyinde ayrışma

pixel vector quantized variational autoencoder (Pixel VQ-VAE)

piksel vektör kuantize edilmiş varyasyonel otomatik kodlayıcı

predictions

tahminler

procedural content generation (PCG)

prosedürel içerik üretimi (PIÜ)

procedural content generation via machine learning (PCGML)

makine öğrenimi ile prosedürel içerik üretimi (MÖPIÜ)

query matrix

sorgu matrisi

receptive field
algılama alanı

reconstruction loss
yeniden üretim kaybı

reinforcement learning
pekiştirmeli öğrenme

residual network (ResNet)
artık sinir ağı

saturation (S)
doygunluk

self attention map
öz dikkat haritası

skip connections
bağlantıları atlama

sprite
hareketli çizim

star generative adversarial networks (StarGAN)
yıldız çekişmeli üretici ağ

stochastic gradient descent (SGD)
stokastik gradyan inişi (SGİ)

stop gradient function
gradyan durduran fonksiyon

stride
adım

style blocks
stil blokları

style label
stil etiketi

style generative adversarial network (Style GAN)
stil çekişmeli üretici ağ

style generative adversarial network 2 (Style GAN 2)
stil çekişmeli üretici ağ 2

trapped ball segmentation
küresel hapsedme segmentasyonu

upsample
üst örnekleme

value (V)

değer

value matrix

değer matrisi

variational autoencoder (VAE)

varyasyonel otomatik kodlayıcı (VOK)

vector quantization layer

vektör kuantizasyon katmanı

vector quantized variational encoder (VQ-VAE)

vektör kuantize edilmiş varyasyonel otomatik kodlayıcı (VK-VOK)

zero-sum game

sıfır toplamlı oyun



TEZDEN ÜRETİLMİŞ YAYINLAR

Konferans Bildirisi

1. V. C. Bacaksız ve O. Altun, "Enhancing Automated Pixel Art Character Creation with DCGAN," in *Proc. 4th Int. Cankaya Sci. Studies Congr.*, Ankara, Türkiye, 2024, ss. 50-61. erişim adresi: https://www.izdas.org/_files/ugd/d0a9b7_54d8893211274c9b8be17ccba7304266.pdf

