

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**360 Artificial Intelligence Methods (3+0)**

**2023–2024 SPRING SEMESTER**

**PROJECT-2 REPORT**

**DELIVERY DATE**

20/05/2024

**PREPARED BY**

05210000195, Muhammed Said Golotooğlu

İçindekiler

[1) Projenin Tanımı 2](#_Toc167142367)

[2) Araştırma (Ön çalışma) 2](#_Toc167142368)

[3) Kullanılan Ortam, Yöntem ve Kütüphaneler 3](#_Toc167142369)

[4) Önerilen (Geliştirilen / Kullanılan) Yöntem 3](#_Toc167142370)

[5) Deneysel Çalışmalar 8](#_Toc167142371)

[6) Sonuç 9](#_Toc167142372)

[7) Ek 1: Başarım İyileştirme 10](#_Toc167142373)

[8)Ek 2: Literatür Katkısı 10](#_Toc167142374)

[9) Ek 3:Faydanılan Kaynaklar 10](#_Toc167142375)

[10) Ek 4: İş Paketleri ve İşbölümü 11](#_Toc167142376)

[11) Öz değerlendirme Tablosu 11](#_Toc167142377)

# 1) Projenin Tanımı

Proje 2 kapsamında iki tane proje ile uğraştım. İlk projemde çözmeyi amaçladığım problem şuydu: bir derin öğrenme modeli geliştireceğiz ama elimizde o alanla ilgili az çeşitli ve az miktarda veri var. GAN yardımıyla fake veriler üretip veri setimizi arttırıp çeşitlendireceğiz. Örnek olarak GAN üç tane veri seti (cifar10, pigs, mnıst) ile ayrı ayrı eğitildi. İkinci projemde ise knapsack problemini genetik algoritma ile çözdüm.

# 2) Araştırma (Ön çalışma)

Araştırmalarım sonucu, GAN mimarisi ile generator ve discriminator yapılarını öğrendim. Generator ve discriminator’deki katmanları optimize etmek ve boyutularını doğru ayarlamak en zor işti, bu yüzden öğrenmek için en fazla zaman ayırdığım bölüm burasıydı. Kullandığım bazı katmanların (UpSampling2D, Conv2D, Conv2DTranspose) farklılıklarını öğrendim. Ayrıca GAN’ın benzer görüntüler oluşturma, görüntülerin yanlış ve eksik olması, yüksek epoch sayısına rağmen karmaşık görüntüler gibi sorunlar için ekstra araştırma yaptım. İkinci projem için ise knapsack problemini araştırdım ve farklı selection algoritmalarına baktım.

<https://www.youtube.com/watch?v=Sw9r8CL98N0>

<https://www.youtube.com/watch?v=TpMIssRdhco>

<https://www.youtube.com/watch?v=afQpqI3K6Cc>

<https://www.youtube.com/watch?v=YtOCygVEejo&t=14s>

<https://stackoverflow.com/questions/68904476/after-some-number-of-epochs-fake-image-creation-become-worst-in-gan>

<https://stackoverflow.com/questions/53654310/what-is-the-difference-between-upsampling2d-and-conv2dtranspose-functions-in-ker>

<https://medium.com/@marcodelpra/generative-adversarial-networks-dba10e1b4424#:~:text=Architecture&text=A%20Generative%20Adversarial%20Network%20(GAN,produces%20data%20(like%20images>).

<https://github.com/soumith/ganhacks>

<https://bilgisayarkavramlari.com/2008/03/24/torba-problemi-knapsack-problem/>

<https://www.javatpoint.com/0-1-knapsack-problem#:~:text=The%200%2F1%20knapsack%20problem%20means%20that%20the%20items%20are,pick%20the%202kg%20item%20completely>.

# 3) Kullanılan Ortam, Yöntem ve Kütüphaneler

Programlama dili olarak ilk projem için Python (3.11.7) ikincisi için ise Java’yı (19.0.2) tercih ettim. Python’da kütüphane olarak PyTorch, Keras, TensorFlow, matplotlib, os, cv2, numpy, … kullandım. Yöntem olarak ise ilk projemde yapay sinir ağları, CNN, GAN; ikinci projemde genetik algoritma kullandım.

# 4) Önerilen (Geliştirilen / Kullanılan) Yöntem

(DL / ML / Yapay Zeka Yöntemlerinin Yapısı veya Modeli, varsa diğer özellikleri açıklanacak, *Yönteminizin Şeması veya Algoritması verilecek*) [varsa Öznitelik Çıkarma Teknikleri]

İki projeyle uğraştığımdan ve birinci projede iki ayrı kod olduğundan detaylı anlatım ve bilgilendirme için lütfen readme.txt dosyasını okuyunuz.

Proje 1, GAN\_model2 hakkında açıklama:

GAN\_model2, bir Generative Adversarial Network (GAN) uygulamasıdır ve iki ana bileşenden oluşur: Generator ve Discriminator. Bu model, DL (Derin Öğrenme) kapsamında bir GAN modelini oluşturur ve eğitir. Modelin ayrıntılı açıklaması:

Generator (Üreteç):

Yapısı: Latent (gizli) bir uzaydan alınan rastgele vektörleri (gürültü) anlamlı görüntülere dönüştürmek için kullanılır. Yapısında tam bağlantılı katmanlar (fully connected layers) ve evrişimli katmanlar (convolutional layers) bulunur.

İnput: Latent uzaydan gelen rastgele bir vektör (genellikle boyutu 100 olarak seçilir).

Output: Üretilen görüntüler (32x32 boyutunda, 3 kanal RGB).

Modelin Katmanları:

Linear Layer: Gizli vektörü (latent vector) 128 x 8 x 8 boyutunda bir tensöre dönüştürür.

ReLU Activation: Aktivasyon fonksiyonu olarak ReLU kullanılır.

Unflatten: Tek boyutlu vektörü (flattened vector) 128 x 8 x 8 boyutunda 4D bir tensöre dönüştürür.

Upsample: Görüntüyü 2 kat büyütür (8x8'den 16x16'ya).

Conv2d + BatchNorm + ReLU: Evrişim katmanı (128 filtre, 3x3 kernel), ardından batch normalization ve ReLU aktivasyonu.

Upsample: Görüntüyü tekrar 2 kat büyütür (16x16'dan 32x32'ye).

Conv2d + BatchNorm + ReLU: Evrişim katmanı (64 filtre, 3x3 kernel), ardından batch normalization ve ReLU aktivasyonu.

Conv2d + Tanh: Son evrişim katmanı (3 filtre, 3x3 kernel), ardından Tanh aktivasyonu.

Öznitelik Çıkarma: Generator, gizli vektörü anlamlı öznitelikler taşıyan bir görüntüye dönüştürür. Bu süreçte her katman, öznitelik çıkarma işlemi yapar.

Discriminator (Ayrımcı):

Yapısı: Üretilen veya gerçek görüntülerin sahte veya gerçek olduğunu belirlemek için kullanılır. Evrişimli katmanlar kullanarak görüntüleri analiz eder.

İnput: 32x32 boyutunda, 3 kanallı (RGB) görüntüler.

Output: Bir skaler değer (gerçek veya sahte olma olasılığı).

Modelin Katmanları:

Conv2d + LeakyReLU + Dropout: Evrişim katmanı (32 filtre, 3x3 kernel), ardından LeakyReLU aktivasyonu ve dropout.

Conv2d + ZeroPad2d + BatchNorm + LeakyReLU + Dropout: Evrişim katmanı (64 filtre, 3x3 kernel), ardından ZeroPad2d, batch normalization, LeakyReLU aktivasyonu ve dropout.

Conv2d + BatchNorm + LeakyReLU + Dropout: Evrişim katmanı (128 filtre, 3x3 kernel), ardından batch normalization, LeakyReLU aktivasyonu ve dropout.

Conv2d + BatchNorm + LeakyReLU + Dropout: Evrişim katmanı (256 filtre, 3x3 kernel), ardından batch normalization, LeakyReLU aktivasyonu ve dropout.

Flatten + Linear + Sigmoid: Flatten katmanı, tam bağlantılı katman (25655=6400 boyutlu), ardından Sigmoid aktivasyonu.

Öznitelik Çıkarma: Discriminator, görüntülerdeki öznitelikleri çıkararak, gerçek ve sahte görüntüleri ayırt etmeye çalışır. Her evrişim katmanı, görüntülerin özniteliklerini analiz eder.

|  |
| --- |
| import torch.optim as optim import torch.nn as nn import torchvision import torch import os  from torchvision import datasets, transforms from torchvision.utils import save\_image  device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  transform = transforms.Compose([  transforms.Resize((32, 32)),  transforms.ToTensor(),  transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)), ])  train\_dataset = datasets.ImageFolder(root="C:\\Users\\Can\\Desktop\\Pigs Dataset (924 images version)", transform=transform) dataloader = torch.utils.data.DataLoader(train\_dataset, batch\_size=16, shuffle=True)  os.makedirs("images", exist\_ok=True)   class Generator(nn.Module):  def \_\_init\_\_(self, latent\_dim):  super(Generator, self).\_\_init\_\_()   self.model = nn.Sequential(  nn.Linear(latent\_dim, 128 \* 8 \* 8),  nn.ReLU(),  nn.Unflatten(1, (128, 8, 8)),  nn.Upsample(scale\_factor=2),  nn.Conv2d(128, 128, kernel\_size=3, padding=1),  nn.BatchNorm2d(128, momentum=0.78),  nn.ReLU(),  nn.Upsample(scale\_factor=2),  nn.Conv2d(128, 64, kernel\_size=3, padding=1),  nn.BatchNorm2d(64, momentum=0.78),  nn.ReLU(),  nn.Conv2d(64, 3, kernel\_size=3, padding=1),  nn.Tanh()  )   def forward(self, z):  img = self.model(z)  return img   class Discriminator(nn.Module):  def \_\_init\_\_(self):  super(Discriminator, self).\_\_init\_\_()   self.model = nn.Sequential(  nn.Conv2d(3, 32, kernel\_size=3, stride=2, padding=1),  nn.LeakyReLU(0.2),  nn.Dropout(0.25),  nn.Conv2d(32, 64, kernel\_size=3, stride=2, padding=1),  nn.ZeroPad2d((0, 1, 0, 1)),  nn.BatchNorm2d(64, momentum=0.82),  nn.LeakyReLU(0.25),  nn.Dropout(0.25),  nn.Conv2d(64, 128, kernel\_size=3, stride=2, padding=1),  nn.BatchNorm2d(128, momentum=0.82),  nn.LeakyReLU(0.2),  nn.Dropout(0.25),  nn.Conv2d(128, 256, kernel\_size=3, stride=1, padding=1),  nn.BatchNorm2d(256, momentum=0.8),  nn.LeakyReLU(0.25),  nn.Dropout(0.25),  nn.Flatten(),  nn.Linear(256 \* 5 \* 5, 1),  nn.Sigmoid()  )   def forward(self, img):  validity = self.model(img)  return validity   def train(num\_epochs, latent\_dim, optimizer\_d, optimizer\_g, adversarial\_loss, generator, discriminator):   for epoch in range(num\_epochs):  for i, (images, \_) in enumerate(dataloader):   valid = torch.ones(images.size(0), 1, device=device)  fake = torch.zeros(images.size(0), 1, device=device)   images = images.to(device)   # Train Discriminator   optimizer\_d.zero\_grad()  z = torch.randn(images.size(0), latent\_dim, device=device)  fake\_images = generator(z)   real\_loss = adversarial\_loss(discriminator(images), valid)  fake\_loss = adversarial\_loss(discriminator(fake\_images.detach()), fake)  d\_loss = (real\_loss + fake\_loss) / 2   d\_loss.backward()  optimizer\_d.step()   # Train Generator   optimizer\_g.zero\_grad()  gen\_images = generator(z)   g\_loss = adversarial\_loss(discriminator(gen\_images), valid)  g\_loss.backward()  optimizer\_g.step()   if (i + 1) % 10 == 0:  print(  f"Epoch [{epoch + 1}/{num\_epochs}] Batch {i + 1}/{len(dataloader)} "  f"Discriminator Loss: {d\_loss.item():.4f} "  f"Generator Loss: {g\_loss.item():.4f}"  )   if (epoch + 1) % 10 == 0:  with torch.no\_grad():  z = torch.randn(16, latent\_dim, device=device)  generated = generator(z).detach().cpu()  grid = torchvision.utils.make\_grid(generated, nrow=4, normalize=True)  # Save the generated images  save\_image(grid, f"images10/epoch\_{epoch + 1}.png")   def main():  latent\_dim = 100  learning\_rate = 0.0002  num\_epochs = 1000   generator = Generator(latent\_dim).to(device)  discriminator = Discriminator().to(device)   adversarial\_loss = nn.BCELoss()   optimizer\_g = optim.Adam(generator.parameters(), lr=learning\_rate)  optimizer\_d = optim.Adam(discriminator.parameters(), lr=learning\_rate)   train(num\_epochs, latent\_dim, optimizer\_d, optimizer\_g, adversarial\_loss, generator, discriminator)   main() |

# 5) Deneysel Çalışmalar

İlk projedeki verisetlerini aşağıda verdiğim linkler üzerinden aldım domuzlarla alakalı olanları tek bir kümede topladım. İlk üç veriseti domuz fotoğrafları hakkında ve sırasıyla 28, 20, 898 veri içeriyorlar. Tek sınıfa sahipler (pigs). GAN eğitimi için test verisine gerek yoktur, tüm verilerle discriminator eğitilir ve random noise ile generator’un ürettiği verilere discriminator tarafından real ya da fake etiketi verilir. Detaylı anlatım için lütfen readme.txt dosyasını okuyunuz. Diğer verisetleri ise cifar10 ve mnıst sırasıyla şöyle özetlenmiştir:

CIFAR-10 Veri Seti

Konusu: CIFAR-10, çeşitli sınıflardan 32x32 boyutunda renkli (RGB) görüntülerden oluşan bir veri setidir. Görüntüler 10 farklı sınıfa aittir ve her sınıf, dünya genelindeki nesneleri temsil eder. Veri seti genellikle görüntü sınıflandırma algoritmalarını eğitmek ve değerlendirmek için kullanılır.

Öznitelik Sayısı: Her görüntü 32x32 piksel ve 3 renk kanalı (RGB) içerir. Bu nedenle, her bir görüntü toplamda 32 x 32 x 3 = 3072 öznitelikten oluşur.

Sınıf Sayısı: 10 sınıf vardır. Bu sınıflar: Airplane (Uçak), Automobile (Otomobil), Bird (Kuş), Cat (Kedi), Deer (Geyik), Dog (Köpek), Frog (Kurbağa), Horse (At), Ship (Gemi), Truck (Kamyon)

Örnek Sayısı: CIFAR-10 veri seti toplamda 60,000 görüntü içerir: 50,000 eğitim ve 10,000 test görüntüsü.

MNIST Veri Seti

Konusu: MNIST (Modified National Institute of Standards and Technology) veri seti, el yazısı rakamların (0'dan 9'a kadar) 28x28 piksel boyutunda gri tonlamalı (grayscale) görüntülerini içerir. Veri seti, makine öğrenme ve derin öğrenme modellerinin eğitimi ve değerlendirilmesi için yaygın olarak kullanılır.

Öznitelik Sayısı: Her görüntü 28x28 piksel içerir, bu nedenle her bir görüntü 28 x 28 = 784 öznitelikten oluşur.

Sınıf Sayısı: 10 sınıf vardır (0'dan 9'a kadar olan rakamlar).

Örnek Sayısı: MNIST veri seti toplamda 70,000 görüntü içerir: 60,000 eğitim ve 10,000 test görüntüsü.

<https://www.kaggle.com/datasets/trainingdatapro/farm-animals-pigs-detection-dataset?select=boxes>

<https://www.kaggle.com/datasets/eljoenaimuninga/pigs-dataset>

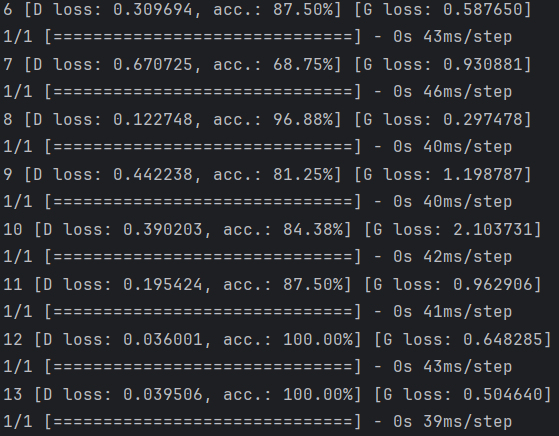
<https://images.cv/dataset/pig-image-classification-dataset>

<https://www.cs.toronto.edu/~kriz/cifar.html>

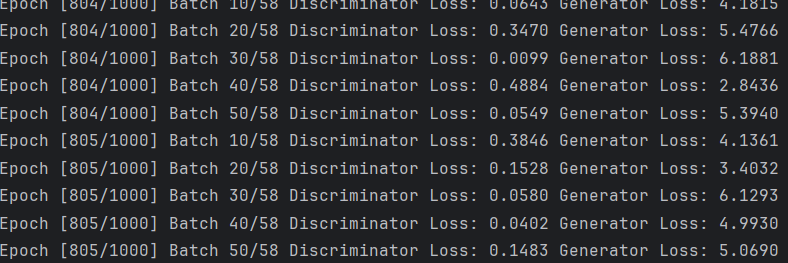
<https://www.kaggle.com/datasets/hojjatk/mnist-dataset>

örnek ss’ler:

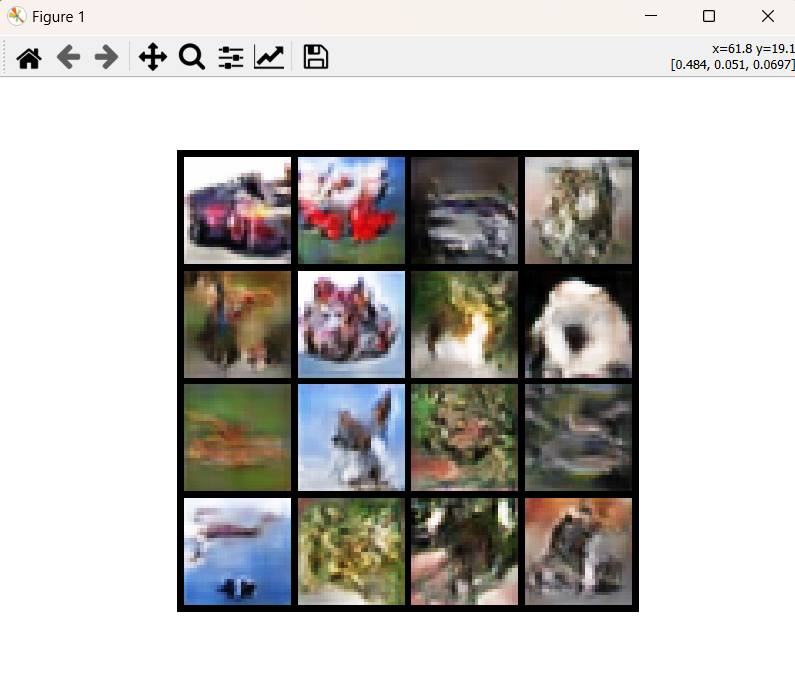
1. GAN\_model1



2. GAN\_model2



3. Other Model epoch=25



# 6) Sonuç

i. Projem başarılı oldu bence, daha iyi veri seti ve gelişmiş cihazlar ile (epoch sayısı arttıkça bilgisayarım çok yavaşladı) daha iyi olabilirdi.

ii. Gerçek hayatta özellikle tıp alanında kullanılabilir. Mesela çok ender bir hastalığın röntgen üzerinden tespiti için eğitilecek modele çeşitli ve yeterli sayıda veri sağlanımı için kullanılabilir. Doktorlar ve tıp alanındaki uzmanların (yazılımcı, araştırmacı) oldukça işine yarayacaktır.

iii. Projem sayesinde az sayıda ve çeşitsiz veri içeren problemlerin çözümünde aşırı verimli ve hızlı ilerleyebiliriz. Bu da ekonomik faydaya sebep olur ama en önemlisi tıp alanında kullanımı insan ve hayvan sağlığı alanında oldukça yararlı olacaktır.

iv. Bu projelere çalışırken özellikle model eğitme ve yapay zekâ alanlarında işlerin nasıl yürüdüğüne dair büyük bir bakış açısı kazandım. Kendim de bu alanda çalışmak istediğimden artık önümü daha rahat görebiliyorum. Verileri ön işleme ve manipüle etme konusunda tecrübe kazandım. Teknik olarak yapay sinir ağları, generative advarsarial networks, generator, discriminator, aktivasyon fonksiyonları, back propogation, gradient descent, gradient vanishing, overfitting, convolution, pattern detection, unsupervised learnign, reinforcement learning, genetic algorithm, seleciton algorithm vb. gibi konuları öğrendim. İleride bu alanda çalışıp verilerle oynamak ve yeni modeller oluşturmak istiyorum. Ayrıca bu çalışmalardan sonra günlük hayatta makinelerin nasıl çalıştığını, YouTube öne çıkarma algoritmasını, bana sunulan reklamların nasıl beni tanıdığını daha iyi fark edebiliyorum.

# 7) Ek 1: Başarım İyileştirme

İlk projemde başarımı arttırmak için generator ve discriminator yapısını sürekli güncelledim. Katmanların sayısını arttırdım, azalttım, farklı katmanlar ekledim, aktivasyon fonksiyonlarını değiştirdim (relu, leakly relu, tanh, sigmoid), parametrelerini (kernel\_size, stride, padding) değiştirdim ve veri setindeki görsellerin boyutları ile oynadım 32\*32 ve 64\*64 boyutlarını denedim. İkinci projemde ise ağırlıkları ve değerlerini güncelledim farklı sonuçlar için ve seleciton kısmını yeniledim. Bu değişiklikler sonucu ilk projemde 4-5 katman (UpSampling2D, Conv2D, Conv2DTranspose), dropout (0.25) ve batch normalization (64, momentum=0.78) katmanları, kernel\_size = 3, stride = 2, paddimg = ‘same’, görsel boyutu = 64\*64 değerleri ile en verimli sonucu aldığımı gördüm. İkinci projemde ise ağırlık ve değer parametreleri düzgün ayarlandığında başarı oranının %55 civarında arttığını saptadım. Özetlemek gerekirse; eğer iki çözümün total değeri yakınsa genetik algoritma yeterli doğal seçilim yapamıyor ve başarı oranı düşüyor, eğer fark fazlaysa düşük olanlar hemen eleniyor ve ileri nesillerde asıl çözüm kalıyor.

# 8)Ek 2: Literatür Katkısı

Bu alandaki kaynakları taradığımda çoğu yöntemin benzer olduğunu gördüm. Benim farkım veri sayısını arttırmak ve çeşitlendirmek için GAN kullanmak ve kullandığım GAN’daki katman yapısı ve sayısını en iyi sonuç için optimize etmektir.

# 9) Ek 3:Faydanılan Kaynaklar

Faydalandığım kaynakların bir çoğunu yukarıda saydım, kod olarak şu kodlardan yararlandım:

1. https://github.com/prajwaldp/gan-image-colorizer/blob/master/GAN.ipynb

2. https://www.wouterbulten.nl/posts/getting-started-with-gans-2-colorful-mnist/

3. https://www.kaggle.com/code/peremartramanonellas/gan-tutorial-2-generating-color-images

Farklılığım şu: Katman sayısı ve niteliklerini değiştirdim ve ayrıca az veri bulunan çeşitsiz veri setlerini çeşitlendirip sayısını arttırmak için GAN yapısını kullandım.

# 10) Ek 4: İş Paketleri ve İşbölümü

Tek kişi olduğumdan projedeki tüm işleri ben yaptım.

# 11) Öz değerlendirme Tablosu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **İstenen Madde** | **Var** | **Açıklama** | **Tahmini Not** |
| **1** | **Kapak Sayfası, Problemin Tanımı, Kullanılan Ortam, Yöntem ve Kütüphaneler, Araştırma (10)** |  | Araştırma dışındaki seçenekleri kendi fikirlerim doğrultusunda yazdım ve diğer bilgiler için ise Python ve java’yı kontrol ettim. Araştırmayı ise chat-gpt ve google’a sorarak hallettim. | 10 |
| **2** | **Önerilen Yöntem (10)** |  | Yöntem olarak ilk projemde GAN mimarisini ikinci projemde ise genetik algoritmayı kullandım. | 9 |
| **3** | **Deneysel Çalışmalar (10)** |  | Bu aşama için tüm kodlara loss değerlerini yazdıracak kısımlar ekledim. Ayrıca veri setlerimi zaten detaylı incelediğimden kalan tüm bilgileri kolayca doldurdum. Eğitim ve test kısmı ise GAN’larda farklı çalıştığından ilgili yerlerde bunu belirttim. | 8 |
| **4** | **Proje Rapor Biçimi, Organizasyonu, Boyutu, Kalitesi, Kaynakça ve atıflar (10)** |  | Rapor kısmı zaten hazırdı, kaynakları ise arama geçmişimden yararlanarak yazdım. | 8 |
| **5** | **Sonuç (10)** |  | Kendi düşüncelerim doğrultusunda yazdım. | 10 |
| **6** | **Ek 1: Başarım İyileştirme (10)** |  | Kodlarda yaptığım değişiklikler sonucu gözlemlediğim olguları zaten not etmiştim. Rapora aktardım. | 10 |
| **7** | **Ek 2: Literatür Katkısı (10)** |  | Google ile araştırma yaptım ve not ettiklerimi yazdım. | 10 |
| **8** | **Ek 3: Faydanılan Kaynaklar (10)** |  | Faydalandığım kaynakları zaten geçmişimden bakarak not etmiştim oradan yararlanarak doldurdum. | 10 |
| **9** | **Ek 4: İş Paketleri ve İşbölümü (10)** |  | Tek kişi yaptığından yazmak pek zor olmadı. | 10 |
| **10** | **Özdeğerlendirme Tablosu (10)** |  | En basit kısım, sadece yaptım 😊 | 10 |
| **100 üzerinden Toplam Not:** | | | | 95 |

**Proje <= 4 kişiye kadar ortak teslim** edilebilir. Herşey dahil **Min 5 sayfa – Max 13 sayfa**

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır. Tahmini not kısmına da ilgili maddeden kaç almayı beklediğinizi yazmalısınız.**

**Not: Raporu ve kodları sisteme yüklenmeyen proje değerlendirilmemektedir.**