

Çanakkale Onsekiz Mart Üniversitesi, Mühendislik Fakültesi,
Bilgisayar Mühendisliği Akademik Dönem 2022-2023
Ders: BLM-4014 Yapay Sinir Ağları/Bahar Dönemi
ARA SINAV SORU VE CEVAP KAĞIDI
Dersi Veren Öğretim Elemanı: Dr. Öğretim Üyesi Ulya Bayram

Öğrenci Adı Soyadı: Zakaria aljamous
Öğrenci No: 170401085

14 Nisan 2023

Açıklamalar:

- Vizeyi çözüp, üzerinde aynı sorular, sizin cevaplar ve sonuçlar olan versiyonunu bu formatta PDF olarak, Teams üzerinden açtığım assignment kısmına yüklemeniz gerekiyor. Bu bahsi geçen PDF'i oluşturmak için LaTeX kullandıysanız, tex dosyasının da yer aldığı Github linkini de ödevin en başına (aşağı url olarak) eklerseniz bonus 5 Puan! (Tavsiye: Overleaf)
- Çözümlerde ya da çözümlerin kontrolünü yapmada internetten faydalanmak, ChatGPT gibi servisleri kullanmak serbest. Fakat, herkesin çözümü kendi emeğinden oluşmak zorunda. Çözümlerinizi, cevaplarınızı aşağıda belirttiğim tarih ve saate kadar kimseyle paylaşmayınız.
- Kopyayı önlemek için Github repository'lerinizin hiçbirini **14 Nisan 2023, saat 15:00'a kadar halka açık (public) yapmayınız!** (Assignment son yükleme saati 13:00 ama internet bağlantısı sorunları olabilir diye en fazla ekstra 2 saat daha vaktiniz var. **Fakat 13:00 - 15:00 arası yüklemelerden -5 puan!**
- Ek puan almak için sağlayacağınız tüm Github repository'lerini **en geç 15 Nisan 2023 15:00'da halka açık (public) yapmış olun linklerden puan alabilmek için!**
- **14 Nisan 2023, saat 15:00'dan sonra gönderilen vizeler değerlendirilmeye alınmayacak, vize notu olarak 0 (sıfır) verilecektir!** Son anda internet bağlantısı gibi sebeplerden sıfır almayı önlemek için assignment kısmından ara ara çözümlerinizi yükleyebilirsiniz yedekleme için. Verilen son tarih/saatte (14 Nisan 2023, saat 15:00) sistemdeki en son yüklü PDF geçerli olacak.
- Çözümlerin ve kodların size ait ve özgün olup olmadığını kontrol eden bir algoritma kullanılacaktır. Kopya çektiği belirlenen vizeler otomatikman 0 (sıfır) alacaktır. Bu nedenle çözümlerinizi ve kodlarınızı yukarıda sağladığım gün ve saatlere kadar kimseyle paylaşmayınız.
- Bu vizeden alınabilecek en yüksek not 100'dür. Toplam aldığınız puan 100'ü geçerse, aldığınız not 100'e sabitlenecektir.
- LaTeX kullanarak PDF oluşturanlar öz geçmişlerine LaTeX bildiklerini de eklemeyi unutmasınlar :)
- Bu vizedeki soruların çözümleri ve tex dosyası için istediğiniz kadar sayıda Github repository'si oluşturabilirsiniz. Sadece yukarıda belirttiğim tarihlerde (14 Nisan 2023 saat 15:00 ile 15 Nisan 2023 saat 15:00 arasında) public yapmayı/halka açmayı ve Github profilinizi de öz geçmişinize eklemeyi unutmayın :)
- Bilgisayarınıza hiçbir program kurmadan, Overleaf, Google Colab, Kaggle gibi ortamlar üzerinden bu vizeyi tamamlamanız mümkün. İyi çalışmalar!!

(Ekstra 5 puan) <https://github.com/Zakaria-aljamous/yapay-sinir-aglari-desi-COMU/tree/main/odev-pdf.com>

1 (Toplam 10 Puan) Hopfield Networks:

1.1 (2 Puan) ChatGPT'ye size Hopfield Network'ler ile ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT'nin çıkarttığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.

Hopfield Network'ler ile ilgili detaylı bilgi

Hopfield ağları, bir tür yapay sinir ağıdır ve bellek ve öğrenme işlevleri gerçekleştirebilir. Ağ, birbirine bağlı bir dizi nöron içerir ve nöronlar iki durumda olabilir: "açık" veya "kapalı". Ağın amacı, belirli bir durum vektörü verildiğinde, ağın o vektörle uyumlu bir şekilde durumunda sabitlenmesidir. Hopfield ağları, çeşitli uygulamalarda kullanılmaktadır, örneğin görüntü tanıma, örüntü tanıma, optimizasyon problemleri gibi.

Ağın amacı nedir bana bir kaç tane örnek verebilir misin

Ağın amaçlarından biri, belirli bir durum vektörü verildiğinde, ağın o vektörle uyumlu bir şekilde durumunda sabitlenmesidir. Bu, ağın öğrenme işlevinin bir yansımasıdır. Öğrenme süreci, ağın belirli bir veri kümesiyle eğitilmesi yoluyla gerçekleştirilir. Hopfield ağları, çeşitli uygulamalarda kullanılmaktadır. Örneğin, Hopfield ağları, görüntü tanıma ve örüntü tanıma gibi uygulamalarda kullanılabilir. Bunun yanı sıra, Hopfield ağları, yol planlama ve hatta Sudoku gibi oyunlar için çözüm üretme gibi çeşitli optimizasyon problemlerinin çözümünde de kullanılabilir Hopfield ağları, diğer yapay sinir ağı türlerine göre daha az karmaşık olmasına rağmen, yine de belirli uygulamalar için oldukça faydalıdır.

Python kodu ile hopfileNetwork nasıl yapabilirim zor mu !

Hopfield ağlarını Python'da kodlamak oldukça kolaydır. İşte özet bir Python kodu örneği:

```
import numpy as np
```

```
class HopfieldNetwork:
```

```
    def __init__(self, n):
        self.weights = np.zeros((n, n))
        self.threshold = 0
```

```
    def train(self, patterns):
        for pattern in patterns:
            self.weights += np.outer(pattern, pattern)
        np.fill_diagonal(self.weights, 0)
```

```
    def predict(self, pattern):
        while True:
            activation = np.dot(self.weights, pattern) - self.threshold
            new_pattern = np.where(activation >= 0, 1, -1)
            if np.array_equal(pattern, new_pattern):
                return pattern
            pattern = new_pattern
```

Bu kodda, Hopfield ağını tanımlayan HopfieldNetwork sınıfı bulunur. __init__ fonksiyonu, ağın ağırlık matrisini ve eşik değerini tanımlar. train fonksiyonu, ağın öğrenme işlevini gerçekleştirir ve örüntülerin ağırlıklarına yansıtır. predict fonksiyonu ise, ağın tahmin işlevini gerçekleştirir ve verilen bir giriş örüntüsünü, öğrenilen ağırlıklara göre, sabit bir hale getirir.

Bu temel kod, Hopfield ağlarının uygulaması için kullanılabilir. Ancak, ağın daha karmaşık uygulamaları için değiştirilmesi gerekebilir.

1.2 (8 Puan) ChatGPT'nin açıklamalarından ne anladığınızı buraya kendi kelimelerinizle özetleyin.Örneğin "Hopfield Network nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?" gibi. Anlamadığınız kısımlar varsa ve ChatGPT'nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!

Hopfield ağları, yapay zeka alanında geliştirilmiş basit temel sinir ağlarından biridir ve sistemin kendi kendine aldığı verilerden öğrenmesini sağlayan kendi kendine öğrenen ağlar olarak kabul edilir. Aynı zamanda lineer dönüşümlere dayanan ağırlıklı hesaplamaya çok yakın bir yapıya sahip sinir ağlarından biridir.Bu ağlar sinir ağları alanında eski modeller olarak kabul edilir, ancak hala birden fazla uygulamada kullanılmaktadır.

2 (Toplam 10 Puan) Boltzman Machine:

2.1 (2 Puan) ChatGPT'ye size Boltzman Machine ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT'nin çıkarttığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.

Boltzmann Machine, yapay sinir ağları için bir modeldir ve Geoffrey Hinton ve Terry Sejnowski tarafından 1985 yılında geliştirilmiştir. Veri analizi, makine öğrenimi, görüntü ve ses işleme gibi çeşitli alanlarda kullanılır.Boltzmann Machine, birbirine bağlı bir dizi öge içerir ve yalnızca iki durum (0 ve 1) içeren birimlerden oluşur. Bu ağ, birimlerin durumunu belirlemek için rastgele faktörler kullanarak, öğrenme verilerini ağırlık ve birimler arasındaki bağlantıları değiştirerek öğrenir.Boltzmann Machine, termal sistemin fiziksel modeline benzer bir şekilde çalışır ve tüm cihazın en olası durumlarını belirleyen Boltzmann dengelemesine dayanır. Ağ, en olası durumları belirlemek için temel enerji işlevini kullanır ve birimler arasındaki bağlantıların ağırlığını değiştirerek bu işlevi iyileştirir.Boltzmann Machine, diğer sinir ağlarıyla kolayca öğrenilemeyen karmaşık olasılık dağılımlarını öğrenme yeteneğine sahiptir. Genellikle büyük veri setlerinin işlenmesi gereken alanlarda kullanılır ve etkili paralel işleme gerektirir. Boltzmann Machine, Contrastive Divergence veya Persistent Contrastive Divergence gibi algoritmalar kullanılarak maliyet fonksiyonunu azaltarak eğitilir.

2.2 (8 Puan) ChatGPT'nin açıklamalarından ne anladığınızı buraya kendi kelimelerinizle özetleyin.Örneğin "Boltzman Machine nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?" gibi. Anlamadığınız kısımlar varsa ve ChatGPT'nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!

Boltzmann Machine, yapay sinir ağları için bir modeldir ve çeşitli alanlarda kullanılır. Bu ağ, öğrenme verilerini kullanarak öğrenir ve tüm cihazın en olası durumlarını belirleyen Boltzmann dengelemesine dayanır. Boltzmann Machine, diğer sinir ağlarıyla kolayca öğrenilemeyen karmaşık olasılık dağılımlarını öğrenme yeteneğine sahiptir ve genellikle büyük veri setlerinin işlenmesi gereken alanlarda kullanılır. Eğitimi için Contrastive Divergence veya Persistent Contrastive Divergence gibi algoritmalar kullanılır.

3 (Toplam 10 Puan) Markov Assumption ve Markov Chain:

3.1 (2 Puan) ChatGPT'ye size Markov assumption ve Markov Chain ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT'nin çıkarttığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.

Markov assumption (Markov varsayımı), bir olayın yalnızca en son durumuna bağlı olduğunu ve geçmiş durumların gelecekteki durumları etkilemediğini varsayar. Bu varsayım, Markov Chain adı verilen bir olasılık modeline dayanır.Markov Chain, belirli bir durum kümesinden oluşan bir modeldir. Bu durumlar, zamanın ilerlemesiyle birbirine bağlı bir şekilde dönüşür. Her bir durum, gelecekteki durumların olasılıklı bir dağılımını

belirler ve bir sonraki duruma geçmek için belirli bir olasılıkla seçilir. Markov Chain'in temel özelliği, yalnızca bir önceki durumun bilinmesi durumunda gelecekteki durumların tahmin edilebileceği fikrine dayanır. Bu nedenle, Markov Chain, belirli bir durum kümesinin özelliklerini modeller ve önceki durumlara dayanmayan durumlarla ilgili tahminler yapar. Markov Chain, pek çok alanda kullanılır. Örneğin, ekonomi, finans, kimya, fizik, biyoloji ve bilgisayar bilimleri gibi çeşitli alanlarda uygulanabilir. Özellikle, rastgele dönüşümlerle modellenen sistemlerde kullanılır.

Markov Chain, gelecekteki durumların tahmin edilebilirliği açısından faydalı bir modeldir. Ayrıca, matematiksel olarak çözülebilir olması ve uygulama kolaylığı sağlaması nedeniyle yaygın olarak kullanılır.

- 3.2 (8 Puan) ChatGPT'nin açıklamalarından ne anladığınızı buraya kendi kelimeleri-nizle özetleyin.** Örneğin “Markov assumption ve Markov Chain nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?” gibi. **Anlamadığınız kısımlar varsa ve ChatGPT'nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!**

Markov assumption, bir olayın yalnızca en son durumuna bağlı olduğunu ve geçmiş durumların gelecekteki durumları etkilemediğini varsayar. Bu varsayım, Markov Chain adı verilen bir olasılık modeline dayanır. Markov Chain, belirli bir durum kümesinden oluşan bir modeldir ve zamanın ilerlemesiyle birbirine bağlı bir şekilde döner. Markov Chain modeli, önceki durumlara dayanmayan durumlarla ilgili tahminler yaparak belirli bir durum kümesinin özelliklerini modeller. Bu model, pek çok alanda kullanılır ve gelecekteki durumların tahmin edilebilirliği açısından faydalıdır. Gerçek hayatta, finansal piyasalarda, doğal dil işleme sistemlerinde, otomatik konuşma tanıma sistemlerinde ve hava durumu tahmini gibi birçok alanda kullanılır.

4 (Toplam 20 Puan) Feed Forward:

- Forward propagation için, input olarak şu X matrisini verin (tensöre çevirmeyi unutmayın):

$$X = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix}$$
 Satırlar veriler (sample'lar), kolonlar öznitelikler (feature'lar).
- Bir adet hidden layer olsun ve içinde tanh aktivasyon fonksiyonu olsun
- Hidden layer'da 50 nöron olsun
- Bir adet output layer olsun, tek nöronu olsun ve içinde sigmoid aktivasyon fonksiyonu olsun

Tanh fonksiyonu:

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Sigmoid fonksiyonu:

$$f(x) = \frac{1}{1 + \exp(-x)}$$

Pytorch kütüphanesi ile, ama kütüphanenin hazır aktivasyon fonksiyonlarını kullanmadan, formülünü verdiğim iki aktivasyon fonksiyonunun kodunu ikinci haftada yaptığımız gibi kendiniz yazarak bu yapay sinir ağını oluşturun ve aşağıdaki üç soruya cevap verin.

- 4.1 (10 Puan) Yukarıdaki yapay sinir ağını çalıştırmadan önce pytorch için Seed değerini 1 olarak set edin, kodu aşağıdaki kod bloğuna ve altına da sonucu yapıştırın:**

4.2

```

import torch

#randwom seed degieri set ediyoruz
Torch.manual_seed(34)

# aktivasyon fonkiysonları tanımlama
def tanh(x):
    return (torch.exp(x) - torch.exp(-x)) / (torch.exp(x) + torch.exp(-x))

def sigmoid(x):
    return 1 / (1 + torch.exp(-x))

# ağın boyutları tanımlıyoruz
input_size = 3
hidden_size = 50
output_size = 1

# matrisi tanımlıyoruz ve tensore çeviriyoruz float32
X = torch.tensor([[1, 2, 3], [4, 5, 6]], dtype=torch.float32)

# ağırlıkları tanımlama ve biasları
W1 = torch.randn(hidden_size, input_size)
b1 = torch.randn(hidden_size, 1)
W2 = torch.randn(output_size, hidden_size)
b2 = torch.randn(output_size, 1)

# feed forward propagation yapmak
hidden_layer = tanh(torch.mm(W1, X.t()) + b1)
output_layer = sigmoid(torch.mm(W2, hidden_layer) + b2)

# output layeri yazdır
print(output_layer)

```

Cıktı: tensor([[0.9902, 0.9767]])

4.3 (5 Puan) Yukarıdaki yapay sinir ağını çalıştırmadan önce Seed değerini öğrenci numaranız olarak değiştirip, kodu aşağıdaki kod bloğuna ve altına da sonucu yapıştırın:

```

import torch

#randwom seed degieri set ediyoruz
Torch.manual_seed(170401085)

```

Cıktı: tensor([[0.2003, 0.0296]])

4.4 (5 Puan) Kodlarınızın ve sonuçlarınızın olduğu jupyter notebook'un Github repository'sindeki linkini aşağıdaki url kısmının içine yapıştırın. İlk sayfada belirttiğim gün ve saate kadar halka açık (public) olmasın:

https://github.com/Zakaria-aljamous/yapay-sinir-aglari-desi-COMU/blob/main/tensor_matrisi_seed_ogrenciNo_ile.ipynb

https://github.com/Zakaria-aljamous/yapay-sinir-aglari-desi-COMU/blob/main/tensor_matrisi_rastgele%20.ipynb

5 (Toplam 40 Puan) Multilayer Perceptron (MLP):

Bu bölümdeki sorularda benim vize ile beraber paylaştığım Prensesi İyileştir (Cure The Princess) Veri Seti parçaları kullanılacak. Hikaye şöyle (soruyu çözmek için hikaye kısmını okumak zorunda değilsiniz):

Bir zamanlar, çok uzaklarda bir ülkede, ağır bir hastalığa yakalanmış bir prenses yaşamış. Ülkenin kralı ve kraliçesi onu iyileştirmek için ellerinden gelen her şeyi yapmışlar, ancak denedikleri hiçbir çare işe yaramamış.

Yerel bir grup köylü, herhangi bir hastalığı iyileştirmek için gücü olduğu söylenen bir dizi sihirli malzemeden bahsederek kral ve kraliçeye yaklaşmış. Ancak, köylüler kral ile kraliçeyi, bu malzemelerin etkilerinin patlayıcı olabileceği ve son zamanlarda yaşanan kuraklıklar nedeniyle bu malzemelerden sadece birkaçının herhangi bir zamanda bulunabileceği konusunda uyarılmışlar. Ayrıca, sadece deneyimli bir simyacı bu özelliklere sahip patlayıcı ve az bulunan malzemelerin belirli bir kombinasyonunun prensesi iyileştireceğini belirleyebilecekmiş.

Kral ve kraliçe kızlarını kurtarmak için umutsuzlar, bu yüzden ülkedeki en iyi simyacıyı bulmak için yola çıkmışlar. Dağları tepeleri aşmışlar ve nihayet "Yapay Sinir Ağları Uzmanı" olarak bilinen yeni bir sihirli sanatın ustası olarak ün yapmış bir simyacı bulmuşlar.

Simyacı önce köylülerin iddialarını ve her bir malzemenin alınan miktarlarını, ayrıca iyileşmeye yol açıp açmadığını incelemiş. Simyacı biliyormuş ki bu prensesi iyileştirmek için tek bir şans varmış ve bunu doğru yapmak zorundaymış. (Original source: <https://www.kaggle.com/datasets/unmoved/cure-the-princess>)

(Buradan itibaren ChatGPT ve Dr. Ulya Bayram'a ait hikayenin devamı)

Simyacı, büyülu bileşenlerin farklı kombinasyonlarını analiz etmek ve denemek için günler harcamış. Sonunda birkaç denemenin ardından prensesi iyileştirecek çeşitli karışım kombinasyonları bulmuş ve bunları bir veri setinde toplamış. Daha sonra bu veri setini eğitim, validasyon ve test setleri olarak üç parçaya ayırmış ve bunun üzerinde bir yapay sinir ağı eğiterek kendi yöntemi ile prensesi iyileştirme ihtimalini hesaplamış ve ikna olunca kral ve kraliçeye haber vermiş. Heyecanlı ve umutlu olan kral ve kraliçe, simyacının prensese hazırladığı ilacı vermesine izin vermiş ve ilaç işe yaramış ve prenses hastalığından kurtulmuş.

Kral ve kraliçe, kızlarının hayatını kurtardığı için simyacıya krallıkta kalması ve çalışmalarına devam etmesi için büyük bir araştırma bütçesi ve çok sayıda GPU'su olan bir server vermiş. İyileşen prenses de kendisini iyileştiren yöntemleri öğrenmeye merak salıp, krallıktaki üniversitenin bilgisayar mühendisliği bölümüne girmiş ve mezun olur olmaz da simyacının yanında, onun araştırma grubunda çalışmaya başlamış. Uzun yıllar birlikte krallıktaki insanlara, hayvanlara ve doğaya faydalı olacak yazılımlar geliştirmişler, ve simyacı emekli olduğunda prenses hem araştırma grubunun hem de krallığın lideri olarak hayatına devam etmiş.

Prenses, kendisini iyileştiren veri setini de, gelecekte onların izinden gidecek bilgisayar mühendisi prensler ve prensesler başkalarına faydalı olabilecek yapay sinir ağları oluşturmayı öğretilsinler diye halka açmış ve sınavlarda kullanılmasını salık vermiş."

İki hidden layer'lı bir Multilayer Perceptron (MLP) oluşturun beşinci ve altıncı haftalarda yaptığımız gibi. Hazır aktivasyon fonksiyonlarını kullanmak serbest. İlk hidden layer'da 100, ikinci hidden layer'da 50 nöron olsun. Hidden layer'larda ReLU, output layer'da sigmoid aktivasyonu olsun.

Output layer'da kaç nöron olacağını veri setinden bakıp bulacaksınız. Elbette bu veriye uygun Cross Entropy loss yöntemini uygulayacaksınız. Optimizasyon için Stochastic Gradient Descent yeterli. Epoch sayınızı ve learning rate'i validasyon seti üzerinde denemeler yaparak (loss'lara overfit var mı diye bakarak) kendiniz belirleyeceksiniz. Batch size'ı 16 seçebilirsiniz.

5.1 (10 Puan) Bu MLP'nin pytorch ile yazılmış class'ının kodunu aşağı kod bloğuna yapıştırın:

```
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(13, 100)
        self.fc2 = nn.Linear(100, 50)
        self.fc3 = nn.Linear(50, 1)
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.sigmoid(self.fc3(x))
        return x
```

5.2 (10 Puan) SEED=öğrenci numaranız set ettikten sonra altıncı haftada yazdığımız gibi training batch'lerinden eğitim loss'ları, validation batch'lerinden validasyon loss değerlerini hesaplayan kodu aşağıdaki kod bloğuna yapıştırın ve çıkan figürü de alta ekleyin.

```
# her dönem için veri kümesinde döngü yapın ve eğitim ve doğrulama
kayıplarını hesaplayın
for epoch in range(50):

    train_loss = train_epoch(model, train_loader, criterion, optimizer)
    val_loss = val_epoch(model, val_loader, criterion)

    # eğitim ve doğrulama kayıplarını ilgili listelerine ekleyin
    train_losses.append(train_loss)
    val_losses.append(val_loss)

    # test kaybını hesaplayın ve test kaybı listesine ekleyin
    test_loss = val_epoch(model, test_loader, criterion)
    test_losses.append(test_loss)

    # her dönem için eğitim ve doğrulama kaybını yazdır
    print(f'Epoch {epoch + 1}: Train Loss: {train_loss:.3f}, Val Loss: {val_loss:.3f}, Test Loss: {test_loss:.3f}')

# eğitim, doğrulama ve test kayıplarını çizin
plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.plot(test_losses, label='Testing Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

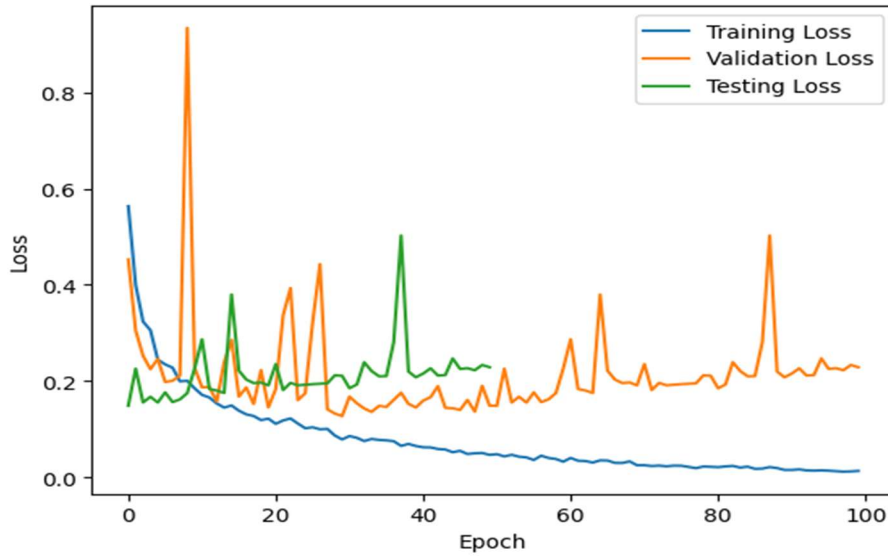
# test kaybını hesaplayın ve test kaybı listesine ekleyin
test_loss = val_epoch(model, test_loader, criterion)
test_losses.append(test_loss)

# her dönem için eğitim ve doğrulama kaybını yazdır
print(f'Epoch {epoch + 1}: Train Loss: {train_loss:.3f}, Val Loss: {val_loss:.3f}, Test Loss: {test_loss:.3f}')

# eğitim, doğrulama ve test kayıplarını çizidir
plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.plot(test_losses, label='Testing Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Çıktı:

```
Epoch 49: Train Loss: 0.012, Val Loss: 0.233, Test Loss: 0.233
Epoch 50: Train Loss: 0.013, Val Loss: 0.229, Test Loss: 0.229
```

5.3 (10 Puan) SEED=öğrenci numaranız set ettikten sonra altıncı haftada ödev olarak verdiğim gibi earlystopping'deki en iyi modeli kullanarak, Prensesi İyileştir test setinden accuracy, F1, precision ve recall değerlerini hesaplayan kodu yazın ve sonucu da aşağı yapıştırın. %80'den fazla başarı bekliyorum test setinden. Daha düşükse başarı oranınız, nerede hata yaptığınızı bulmaya çalışın. %90'dan fazla başarı almak mümkün (ben dedim).

```
# Define the model and load the best state dict obtained through early stopping
model = MLP()
best_model_path = 'best_model.pth'
model.load_state_dict(torch.load(best_model_path))

# Define a function to calculate evaluation metrics
def evaluate(model, dataloader):
    model.eval()
    with torch.no_grad():
        true_positives = 0
        false_positives = 0
        false_negatives = 0
        true_negatives = 0
        for inputs, labels in dataloader:
            outputs = model(inputs)
            predicted = (outputs > 0.5).float()
            true_positives += ((predicted == 1) & (labels == 1)).sum().item()
            false_positives += ((predicted == 1) & (labels == 0)).sum().item()
            false_negatives += ((predicted == 0) & (labels == 1)).sum().item()
            true_negatives += ((predicted == 0) & (labels == 0)).sum().item()

        accuracy = (true_positives + true_negatives) / (true_positives + false_positives + false_negatives + true_negatives)
        precision = true_positives / (true_positives + false_positives)
        recall = true_positives / (true_positives + false_negatives)
        f1 = 2 * (precision * recall) / (precision + recall)

    return accuracy, f1, precision, recall

# Calculate evaluation metrics on the test set
test_accuracy, test_f1, test_precision, test_recall = evaluate(model, test_loader)

# Print the evaluation metrics
print(f'Test Accuracy: {test_accuracy:.4f}')
print(f'Test F1 Score: {test_f1:.4f}')
print(f'Test Precision: {test_precision:.4f}')
print(f'Test Recall: {test_recall:.4f}')
```

Sonuç: Test Accuracy: 0.4650, Test F1 Score: 0.6233, Test Precision: 0.4793, Test Recall: 0.8910

5.4 (5 Puan) Tüm kodların CPU’da çalışması ne kadar sürüyor hesaplayın. Sonra to device yöntemini kullanarak modeli ve verileri GPU’ya atıp kodu bir de böyle çalıştırın ve ne kadar sürdüğünü hesaplayın. Süreleri aşağıdaki tabloya koyun. GPU için Google Colab ya da Kaggle’ı kullanabilirsiniz, iki ortam da her hafta saatlerce GPU hakkı veriyor.

Tablo 1: Buraya bir açıklama yazın

Ortam	Süre (saniye)
CPU	6.72 sc
GPU	0.75841

5.5 (3 Puan) Modelin eğitim setine overfit etmesi için elinizden geldiği kadar kodu gereken şekilde değiştirin, validasyon loss’unun açıkça yükselmeye başladığı, training ve validation loss’ları içeren figürü aşağı koyun ve overfit için yaptığınız değişiklikleri aşağı yazın. Overfit, tam bir çanak gibi olmalı ve yükselmeli. Ona göre parametrelerle oynayın.

1-Mevcut kodunuzda, modele uyum sağlamak için aşağıdaki değişiklikleri deneyebiliriz: Birinci katmanda 200 nöron, ikinci katmanda 100 nöron ve üçüncü katmanda 50 nöron gibi katmanlardaki nöron sayısını artırmak.

```
self.fc1 = nn.Linear(13, 200)
self.fc2 = nn.Linear(200, 100)
self.fc3 = nn.Linear(100, 50)
```

2-Toplu normalleştirmeyi kaldırarak ve bırakma oranını veya ağırlık kaybını azaltarak düzenlileştirmeyi azaltıyoruz. Örneğin, toplu normalleştirmeyi kaldırabilir ve bırakma oranını 0,1’e veya ağırlık azalmasını 0,001’e ayarlayabiliriz.

```
self.fc1 = nn.Linear(13, 200)
self.fc2 = nn.Linear(200, 100)
self.fc3 = nn.Linear(100, 50)
self.relu = nn.ReLU()
self.sigmoid = nn.Sigmoid()
self.dropout = nn.Dropout(0.1)
```

3-100 veya 200 dönem gibi dönem sayısını artırıyoruz. Ancak, doğrulama kaybı artarken eğitim kaybı azalabileceğimizden, modele aşırı uyum sağlama konusunda dikkatli olmalıyız.

```
num_epochs = 100
```

5.6 (2 Puan) Beşinci soruya ait tüm kodların ve cevapların olduğu jupyter notebook’un Github linkini aşağıdaki url’e koyun.

<https://github.com/Zakaria-aljamous/yapay-sinir-aglari-desi-COMU/blob/main/mlp2.ipynb>

6 (Toplam 10 Puan)

Bir önceki sorudaki Prensese İyileştir problemindeki yapay sinir ağına seçtiğiniz herhangi iki farklı regülerizasyon yöntemi ekleyin ve aşağıdaki soruları cevaplayın.

6.1 (2 puan) Kodlarda regülarizasyon eklediğiniz kısımları aşağı koyun:

```
l2_lambda = 0.01

# L2 düzenleme terimini hesapla
for epoch in range(num_epochs):
    for i, (inputs, labels) in enumerate(trainloader):
        l2_regularization = torch.tensor(0.0, requires_grad=True)
        for param in model.parameters():
            l2_regularization += torch.sum(torch.square(param))
```

6.2 (2 puan) Test setinden yeni accuracy, F1, precision ve recall değerlerini hesaplayıp aşağı koyun:

```
Epoch 50/50,
Train Loss: 0.6871,
Loss değeri: 0.1488,
Accuracy değeri: 0.5287,
F1 değeri: 0.6783,
Precision değeri: 0.5132,
Recall değeri: 1.0000
```

6.3 (5 puan) Regülarizasyon yöntemi seçimlerinizin sebeplerini ve sonuçlara etkisini yorumlayın:

Regularizasyon yöntemleri, aşırı uyum (overfitting) problemini çözmek için kullanılan bir dizi tekniklerdir. Bu yöntemler ağırlıkları küçültmek, kısıtlamak veya sınırlamak için kullanılır. L1 ve L2 regularization, en yaygın kullanılan tekniklerdir. L1 regularization, ağırlıkların toplam mutlak değerini sınırlarken, L2 regularization, ağırlıkların karesinin toplamını sınırlar. Regularizasyon yöntemleri, modelin genelleştirme yeteneğini artırarak, test verilerinde daha iyi performans göstermesini sağlar. Ancak, aynı zamanda eğitim verilerinde performansı azaltabilir. Doğru bir regularization yöntemi seçmek, modelin başarısını etkileyebilir.

6.4 (1 puan) Sonucun github linkini aşağıya koyun:

<https://github.com/Zakaria-aljamous/yapay-sinir-aglari-desi-COMU/blob/main/regülarizasyon.ipynb>