

Perceptron ile Karakter Tanıma

01/12/2020

BLM4520 – Yapay Sinir Ağlarına Giriş
Doç. Dr. Sırma Yavuz

Ahmet Onur Akman
16011059



İçindekiler

1. Giriş	3
1.2 Problem	3
1.2 Veri	3
2. Yöntem	4
2.1 Ön Bilgilendirme	4
2.2 Yaklaşım	5
2.3 Girdiler	6
2.4 Fonksiyonlar	7
3. Sistem Çıktıları	8
3.1 Denemeler	8
3.2 Eğitim Süreçleri Grafikselsel Gösterimleri	10
4. Sonuç	11
4.1 Ön Bilgilendirme	11
4.2 Gözlem	12
4.2.1 Veri Türü	12
4.2.2 Learning Rule Karşılaştırması	12
4.2.3 Gürültü	13
4.2.4 Learning Ra	14
4.2.5 Aktivasyon Fonksiyonu Eşik Değeri	14
4.3 Çıkarım	15
4.4 Geliştirilebilirlik	16
4.4.1 Problem	16
4.4.2 Veri	16
4.4.3 Sinir Ağı Yapısı	16
4.4.4 Data Augmentation	16

1. Giriş

Bu bölümde bu çalışma ile çözülmeye çalışılan problem, elde edilmek istenen sonuç ve bu süreçte kullanacağımız verinin nitelikleri incelenecektir.

1.1 Problem

Bu çalışma kapsamında, bize verilen kodlanmış karakter çizimlerinin oluşturulacak bir Perceptron yapısı ile sınıflandırılması istenmektedir. Her bir piksel değeri için birer tane giriş nodu içeren bir adet giriş katmanı ve buna karşılık harfin ne olduğunun çıkarımı yapıldığı ve 7 farklı harf için temsilen 7 farklı çıkış noduna sahip bir çıkış katmanı tasarlanacak, her giriş için 7 farklı çıkışa ait bağlantı ve bu bağlantılar için ağırlık değeri atanacaktır. Problemin en temel noktası, giriş nodundan çıkışa bağlanan bu ağırlık değerlerinin makine tarafından doğru değere yakınsanmasını sağlamaktır. Ağırlık değerleri algoritma tarafından düzenlendikçe, programın verilen 21 farklı çizimi 7 farklı karakter olarak sınıflandırma başarısını arttırmak mümkün olacaktır.

Programın bahsedilen ağırlık değerlerini istenen değerlere yakınlaştırması yöntemiyle “öğrenmesi” için bir öğrenme kuralına bağlı kalması gerekmektedir. Bu kurallar yeni ağırlık değerinin tespiti için izlenecek formülizasyonu içermektedir. Bu çalışma kapsamında, Delta ve Perceptron öğrenme kuralları seçenek olarak sunulmuştur.

İleri seviye yapay sinir ağları temelli projelerde, eldeki veriler training, validation ve test olmak üzere 3 farklı sınıfa ayrılıp farklı amaçlarla kullanılır. Ancak verilen bu ödevin seviyesi dolayısı ile böyle bir kısıt istenmemiştir, eldeki verinin tümü hem training, hem validation, hem de test verisi olarak kullanılması istenmiştir.

1.2 Veri

Verilen karakterler, 2 farklı biçimde kodlanmış, A, B, C, D, E, J, K olmak üzere 7 farklı harfin toplamda üçer farklı fontta çizilmesiyle oluşturulmuştur, dolayısıyla elimizdeki çizim sayısı binary ve bipolar formatta toplam 42 tanedir. Bu çizimlerin piksellerinin her biri binary formatta 1 ve 0, bipolar formatta -1, 0, 1 değerlerine sahip olacak şekilde kodlanmıştır. Çizimin olduğu piksel 1 ile, olmadığı piksel 0 (bipolar kodlama için -1) ile temsil edilmiştir. Her font için karakter çiziminin biçimi farklılık göstermektedir. Beklenen durum, makinenin her çıktı değerine bağlı ağırlık değerlerini düzenleyerek çizimlerdeki kritik noktaları ayırt ederek harfin ne olduğunu başarıyla anlayabilmesidir.

2. Yöntem

Bu bölümde programın yapıtaşları, işleyişi ve bu işleyişi sağlamak adına tercih edilen yöntemler sebepleri ile tarif edilecektir.

2.1 Ön Bilgilendirme

Ödevin kapsamı ilk bölümde ana hatlarıyla anlatılmıştı. Çalışmanın yapıldığı süre boyunca, eldeki az sayıdaki çizimin yüksek sayıdaki epoch ile defalarca sinir ağından geçirilmesinin oluşan çıktının güvenilirliğini olumsuz olarak etkilediği, bu sebeple verilen veri / parametre / durma koşulu varyasyonlarının karşılaştırılmasının değerini düşürdüğü gözlemlenmiştir.

Karşılaşılan bu problemin yapay zeka çalışmalarındaki bilinen adı *overfitting*'dir. Bu problem, sinir ağının bir süre sonra verilen veriyi ezberlemesi ve yüksek doğruluk oranına ulaşmasına rağmen daha sonra gösterilen başka bir veri setini doğru bir şekilde değerlendirememesi sonucunu doğuran bir durumdur. Özellikle böylesine az veriyle çalışılan bu ortamda, az sayıdaki bu veriyi onlarca veya yüzlerce kez tekrarlı olarak sinir ağı yapısından geçirmek bu durumun ortaya çıkmasının önünü açmaktadır. Şayet önerilen yöntem ile yapılan denemelerin bir kısmında sinir ağı modelinin çok kısa sürede 100% doğruluğa ulaştığı görülmüştür ancak böylesine kısa sürüp böylesine bir accuracy'e ulaşmış bir modelin başka fontlarda verilen verilere karşı iyi sonuçlar vermediği barizdir.

Bu probleme bir seviye çözüm olması açısından (ödevin tanımının esnek tutulmasından da faydalanarak) ödevin sınırlamalarının dışına çıkmış ve verilen veri kategorileştirilmiştir. Büyük projelerde eldeki büyük boyutlu veri seti sırasıyla sinir ağını eğitmek, sinir ağının her adımdaki ilerlemesini gözlemek ve daha sonra yeteneğini test etmek adına *training*, *validation*, *testing* olmak üzere üçe ayrılmaktadır. Bundan esinlenerek, bu kategorilerin en kritik 2 tanesi olan training ve validation kategorileri eldeki veriden oluşturulmuştur. **Yani eğitim verileri ve her adımda accuracy takibi için kullanılacak veriler birbirinden farklıdır.** Bu üç kategoriden yalnızca iki tanesinin seçilmesinin sebebi eldeki verinin azlığıdır.

Bu doğrultuda Font 1 ve 2 ile yazılmış veriler training, Font 3 verileri ise validation olarak kullanılmıştır. Her ne kadar böyle bir ayırım yaparak daha doğru sonuçlar alınmaya çalışıldıysa da, her iki kategori için de ayrılmış veri miktarının yine de yetersiz büyüklükte olduğu aşikardır.

Modelin eğitimi için 21 adet çizim yetersizken, training verisinin büyüklüğünü 14 çizime indirmek modelin öğrenme sürecinde belli bir değere takılı kalmasını ve validation verisi için yaptığı tahminlerin bir süre sonra aynı doğruluk oranında sabit kalması sonucunu doğuracaktır. Bu sebeple (ödev kapsamının genişletilmesi ile bizden istenen duruma paralel olarak) verilerin her epoch'un başında kullanıcı tarafından belirlenen sayı kadar pikselinin rastgele bozulması yöntemi programa dahil edilmiştir. Böylece eldeki az sayıda veriden daha fazla durum ortaya çıkacak ve sinir ağı modelinin daha fazla sayıda ihtimale karşı eğitilmesi mümkün olacaktır. Bu yöntemin adapte edilmemesi durumunda

oluşacak durumun gözlemi için data modification modu programın başında kapatılabilir ancak bu rapor boyunca sadece random data modification kullanılarak yapılan training'lerin analizi yapılacaktır.

2.2 Yaklaşım

Ödev kapsamında kullanılan programlama dili Python, programlama için seçilen ortam Spyder olmuştur. Programlama dili seçiminde Python'un seçilmesinin en temel nedeni, ödevin ilerleyen haftalarda geliştirilme ihtimali ve Python'un yapay sinir ağları konusunda kütüphane zenginliğidir.

Çalışmanın gerekliliklerini yerine getirebilmek adına, öncelikle algoritmanın elementlerini iyi kavramak gereklidir. Programın öğrenmesini sağlayan en kritik etmen, şüphesiz ki *learning rule* olarak isimlendirilen, ağırlıkları güncelleme işleminin kurallarını belirleyen formülizasyondur. Bu programda iki farklı öğrenme kuralı kullanılabilir kılınmıştır. Bunlardan ilki Perceptron öğrenme kuralıdır. Bu kural doğrultusunda, her bir ağırlığın güncelleştirmesi Denklem 2.1'e göre yapılmaktadır.

Perceptron Weights Adjustment

$$\Delta w = \eta \times d \times x$$

$d \rightarrow$ Predicted output - Desired output

$\eta \rightarrow$ Learning rate, usually less than 1

$x \rightarrow$ Input data

Denklem 2.1: Perceptron Öğrenme Kuralı

Bu çalışmada bu formüle ek olarak bias değeri kullanılmıştır ve sabit bias değeri için bias'a bağlı ağırlığın güncellemesi yapılmıştır. Bu göz önünde bulundurulduğunda, daha kapsamlı olarak gösterim Denklem 2.2'deki gibi olacaktır.

if $y \neq t$

$$w_i(\text{new}) = w_i(\text{old}) + \eta t x_i$$

$$b(\text{new}) = b(\text{old}) + \eta t$$

else

$$w_i(\text{new}) = w_i(\text{old})$$

$$b(\text{new}) = b(\text{old})$$

Denklem 2.2 Perceptron Öğrenme Kuralı

Bu gösterimdeki "y" elde edilen çıkış, "t" hedeflenen değer, "eta" sembolü learning rate gösterimidir.

Aradaki farklılıkları gözlemleyebilmek ve daha iyi analiz yapabilmek adına kullanılan ikinci öğrenme kuralı Delta Kuralı olarak belirlenmiştir. Bu kurala göre ağırlık güncelleme formülizasyonu Denklem 2.3'te verilmiştir.

$$\Delta w = \eta (t - y) x_i$$

Denklem 2.3 Delta Öğrenme Kuralı

Bu gösterimde ağırlık değişiminin learning rate, error ve giriş değerine bağlantılı olduğu gözlemlenmektedir.

Verilen bu iki öğrenme kuralının en büyük farklılığı ağırlık güncellemesinde değer değişimi için referans alınan parametredir. Perceptron kuralı yalnızca istenen sonuca odaklı güncelleme yaparken, Delta kuralı istenen sonuca ne kadar yakın olduğumuzun değerini referans almaktadır.

Program kapsamında grafik çizdirmek için **“matplotlib”** ve verilen kodlamalar içerisinde piksel değerlerini ayrıştırmak amacıyla regular expression kullanıldığı için **“re”** kütüphanelerine ihtiyaç duyulmuştur. Ayrıca rastgele veri bozma işlemi için ortaya çıkan random sayı üretimi ihtiyacı, **“random”** kütüphanesi ile giderilmiştir. Geri kalan kısımlar için sıfırdan kodlama yapılmıştır. Programın işleyişi genel olarak şu şekildedir.

- 1) **Parametrelerin alınması**
- 2) **Verinin okunması**
- 3) **Training süreci**
- 4) **Hedeflenen accuracy'e veya istenen epoch sayısına ulaşıldığında training'in sonlandırılması**
- 5) **Kullanıcı istediği takdirde adım 2'ye dönülmesi veya elde edilen ağırlık değerlerinin doğruluğunun veri seti için verdiği sonuçların testi**

Accuracy ölçümü için ayrılan validation verileri için o anki ağırlık değerlerinin yaptığı tahminler ve hedeflenen değerlerin karşılaştırılmasından oluşan yüzdelik bir sayı değeri kullanılmıştır.

Aktivasyon fonksiyonu için threshold fonksiyonu kullanılmıştır. Threshold fonksiyonu için farklı durumlarda verilen threshold paramteresine göre, [-1, -1*Threhold) aralığı olumsuz, [-1*Threhold, Threhold] aralığı belirsiz, (Threhold, 1] aralığı olumlu olacak şekilde bipolar çıktılı bir yapı elde edilmiştir. Bu olumsuz – belirsiz – olumlu çıktılar sırasıyla -1, 0, 1 değerleri ile sembolize edilmiştir.

2.3 Girdiler

Kullanıcı programı başlattığı zaman training sürecinde ihtiyaç duyulan parametrelerin seçilmesi talep edilmektedir. Bu parametreler, veri tipi, learning rule, data modifiaction, epoch sayısı, learning rate ve threshold fonksiyonu için eşik değeridir. Bunların yanı sıra, trainingin bitmesiyle beraber yapılacak işlem için yönlendirmeler beklenmektedir.

2.4 Fonksiyonlar

getPixels: Dosya okunması ve alınan metinden RegEx ile pixel değerlerinin çekilmesi

trainWeights: Modelin eğitilmesi, istenen değer ve prediciton karşılaştırmasıyla ağırlıkların güncellenmesi

predict: Verilen veri ve ağırlık değerlerine göre çıkış biriminin değerinin bulunması

threshold: Aktivasyon fonksiyonu, bipolar threshold fonksiyonunun verilen değere, verilen threshold değerine göre uygulanması

accuracy: Verilen veriler ve ağırlık değerlerine göre istenen sonuçların ne kadar alınabildiğinin kontrolü, doğruluk oranını yüzde bir sayı olarak döndürür.

normalize: İstendiği takdirde kullanılmak üzere oluşturulmuştur. Verilen bir diziyi, verilen aralıklar arasında normalize eder.

dataModification: Verilen dizi formatındaki veriyi, istenen modifikasyon yoğunluğuna göre modifiye eder. Gürültü oluşturmak için kullanılmıştır.

3. Sistem Çıktıları

Bu bölümde, model üzerine yapılan çeşitli denemelerden bahsedilecek ve bu denemelerin verdiği sonuçlar paylaşılacaktır.

3.1 Denemeler

Sisteme verilecek parametre kombinasyonlarının sayısı çok fazla olduğu için analiz açısından en anlamlı olacak kombinasyonları denemek adına çıktılar üç farklı şekilde gözlemlenecektir. Gösterilen denemelerin nitelikleri ve verdikleri sonuçlar bir sonraki bölümde irdelenecektir.

İlk adımda parametrelerin değişik durumları karşılaştırılarak, elde edilebilecek en iyi sonuca ulaşmaya çalışılacaktır. Kullanıcı tarafından her program başlangıcında verilen bu parametrelerin hangisinin en iyi sonuçları doğurduğu tespit edilmeye çalışılacaktır.

Bu bağlamda yapılmış denemelerin parametreleri ve çıktıları Tablo 3.1’de verilmiştir.

Deneme	Data Type	Learning Rule	Her Epoch’ta Veri Baş Değiştirilecek Piksel Sayısı	Learning Rate	Threshold Value	Final Validation Accuracy	# of Epoch
1	Binary	Perceptron	5	0.05	0.5	93.87%	2000
2	Bipolar	Perceptron	5	0.05	0.5	93.87%	2000
3	Bipolar	Delta	5	0.05	0.5	95.91%	2000
4	Bipolar	Delta	8	0.05	0.5	100.0%	1068
5	Bipolar	Delta	8	0.1	0.5	100.0%	509
6	Bipolar	Delta	8	0.1	0.7	100.0%	515

Tablo 3.1: Değişik Parametre Değerlerine Göre Program Çıktıları

Bu denemeler ışığında veriyi Bipolar şekilde kodlayıp, Delta kuralı ile, gürültü ile çeşitlendirilmiş veri kullanarak eğiterek iyi sonuçlar aldığı görülmektedir. Bu şekilde yapılan deneme sonucunda yapılan test ile programın eldeki tüm verileri tam doğruluk ile ayırt edebildiği görülmüştür. Bunun sebepleri ve tam olarak bize ne ifade etmesi gerektiği bir sonraki kısımda irdelenecektir.

İkinci deneme metodumuzda, parametreleri sistematik olmayan bir şekilde değiştirerek yüksek gürültülü (verinin yarısına kadar gürültü) veri içerisinde modelin davranışı gözlemlenecektir. Aynı zamanda önceki sistematik ilerleyişte birlikte görme şansı bulamadığımız kombinasyonların ve eleme yöntemi dolayısıyla sonuçlarını çok fazla gözlemleyemediğimiz metotların (Perceptron rule gibi) denenmesi sağlanacaktır.

Bu bağlamda yapılmış denemelerin parametreleri ve çıktıları Tablo 3.2’de verilmiştir.

Deneme	Data Type	Learning Rule	Her Epoch'ta Veri Başı Değiştirilecek Piksel Sayısı	Learning Rate	Threshold Value	Final Validation Accuracy	# of Epoch
7	Binary	Delta	20	0.1	0.5	100.0%	382
8	Binary	Perceptron	22	0.1	0.5	100.0%	391
9	Binary	Perceptron	25	0.2	0.5	100.0%	120
10	Bipolar	Perceptron	27	0.01	0.5	100.0%	651
11	Bipolar	Perceptron	30	0.01	0.9	100.0%	465

Tablo 3.2: Yüksek Gürültü ve Değişen Parametre Değerlerine Göre Program Çıktıları

Yapılan bu denemeler sonucunda, modelin yüksek gürültü ile veri zenginleştirmesine iyi sonuçlar verdiği görülmüştür.

Gürültü ile zenginleştirilmiş veri ile modelin hep 100% doğruluğa yakınsaması gözlem yapmamızı kısıtladığı için, üçüncü metotta az gürültü ile eğitilen modelin farklı kodlama – learning rule varyasyonlarına verdiği çıktılar ele alınacaktır. Ayrıca learning rate düşük ve threshold eşiği nispeten yüksek tutularak çok yüksek accuracy'lere hızla ulaşılmasının önüne geçilmeye çalışılacaktır. Böylece hem kodlama türünün hem de learning kurallarının grafik üzerinden daha ayrıntılı karşılaştırması yapılması hedeflenmektedir.

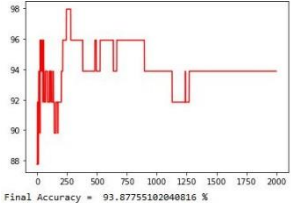
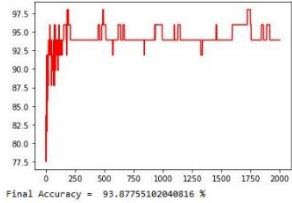
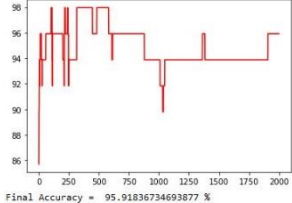
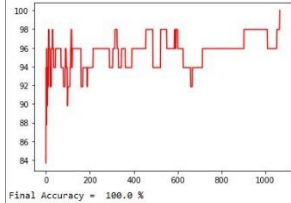
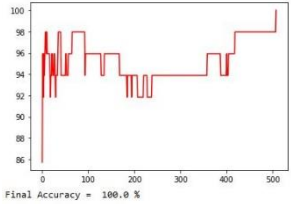
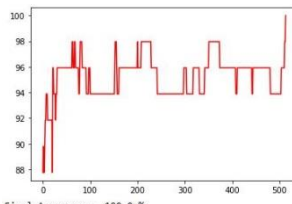
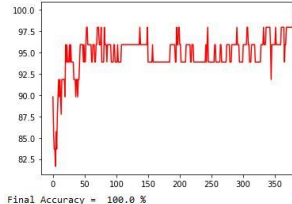
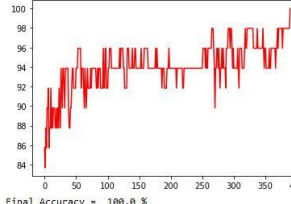
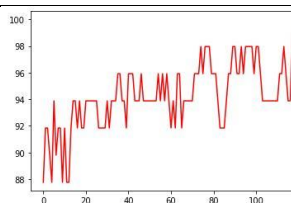
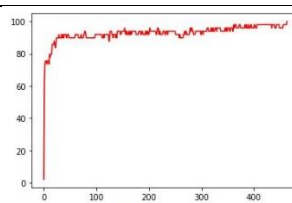
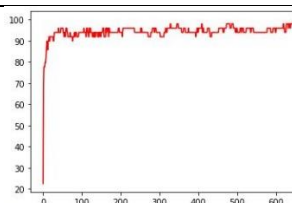
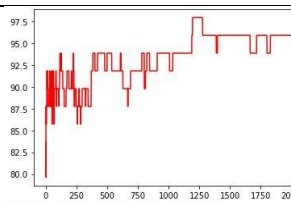
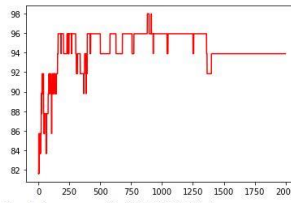
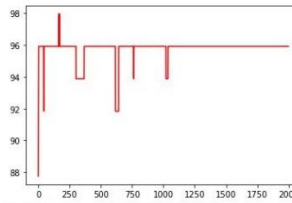
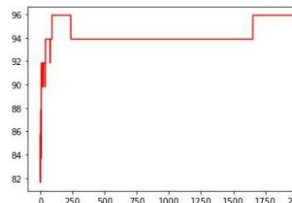
Bu bağlamda yapılmış denemelerin parametreleri ve çıktıları Tablo 3.3'te verilmiştir.

Deneme	Data Type	Learning Rule	Her Epoch'ta Veri Başı Değiştirilecek Piksel Sayısı	Learning Rate	Threshold Value	Final Validation Accuracy	# of Epoch
12	Binary	Delta	2	0.05	0.6	95.91%	2000
13	Binary	Perceptron	2	0.05	0.6	93.87%	2000
14	Bipolar	Delta	2	0.05	0.6	95.91%	2000
15	Bipolar	Perceptron	2	0.05	0.6	95.91%	2000

Tablo 3.3: Düşük Gürültü ve Değişen Parametre Değerlerine Göre Program Çıktıları

3.2 Eğitim Süreçleri Grafikselsel Gösterimleri

Bu bölümde, önceki alt bölümde bahsedilen denemelerin eğitim süreci boyunca tutulmuş validation accuracy değerlerinin training sürecindeki değişimlerinin grafikselsel gösterimi paylaşılmıştır. Bu grafiklerin tam çözünürlükte .JPEG formatındaki hali ödev dosyasına ek olarak paylaşılmıştır.

 Final Accuracy = 93.87755182040816 %	 Final Accuracy = 93.87755182040816 %	 Final Accuracy = 95.91836734693877 %	 Final Accuracy = 100.0 %
Deneme 1	Deneme 2	Deneme 3	Deneme 4
 Final Accuracy = 100.0 %	 Final Accuracy = 100.0 %	 Final Accuracy = 100.0 %	 Final Accuracy = 100.0 %
Deneme 5	Deneme 6	Deneme 7	Deneme 8
 Final Accuracy = 100.0 %	 Final Accuracy = 100.0 %	 Final Accuracy = 100.0 %	 Final Accuracy = 95.91836734693877 %
Deneme 9	Deneme 10	Deneme 11	Deneme 12
 Final Accuracy = 93.87755182040816 %	 Final Accuracy = 95.91836734693877 %	 Final Accuracy = 95.91836734693877 %	
Deneme 13	Deneme 14	Deneme 15	

Tablo 3.4: Deneme Başı Training Süreci Grafikselsel Gösterimleri

4. Sonuç

4.1 Ön Bilgilendirme

Bu çalışma kapsamında eldeki imkanlardan en değerli gözlemi yapmanın uğraşı içerisinde olunmuştur. Buna rağmen eldeki verilerin çok kısıtlı olması, input ve output katmanlarındaki nodların alabileceği değerlerin çeşitliliğinin son derece kısıtlı olması, eğitilmiş modelin gerçek modeller üzerindeki başarısının tespiti amacıyla kullanılacak ekstra verinin olmaması, kullanılan tek *data augmentation* metodunun rastgele gürültü oluşturma olması, eldeki verinin her biriminin çok az sayıda piksel içermesi, sinir ağı modelinin her ne kadar verilen problem için yeterli olsa da yine de artık günümüzde kullanılan sinir ağları modellerine kıyasla yeteneklerinin sınırlı olması gibi sebepler, elde edilen ölçümlerin gerçek hayata kıyasla ne kadar doğruyu gösterdiğini şüpheyi düşürmektedir.

Bahsedilen bu şüphe şu şekilde açıklanabilir, özellikle gürültü düzeyi yüksek olan eğitim metodlarının çok kısa süre içerisinde validation accuracy 100% değerine yakınsadığı görülmüştür. **Bu ve diğer senaryolarda 100% accuracy seviyesine ulaşan sinir ağı modelleri eldeki tüm veriler ile test edilmiş ve gerçekten de eldeki bütün 21 kodlanmış verinin doğru bir şekilde sınıflandırılabilirdiği tespit edilmiştir.** Üstelik bu sınıflandırma, beklenen output'un 1, diğer output nodlarının her seferinde -1 alması gibi keskin bir şekilde meydana gelmiştir. Fakat burada akıllara gelen soru, 500 epoch'tan kısa süren bir eğitim ve 14 çizimlik bir eğitim veri setiyle eğitilmiş bu model, yine aynı piksel sayısına sahip verilen daha farklı fonttaki çizimler için ne kadar başarılı olacaktır? Gerçekten de bu andan itibaren modelin daha önce hiç görmediği, dışarıdan verilen her 7x9'lük çizim, model tarafından doğru bir şekilde sınıflandırılabilir midir?

Bu sorunun kesin bir cevabını bulmak her ne kadar içinde bulunulan durumda mümkün gözükme de, bu eğitim şartlarında böylesine doğrulukta çalışan bir modelin oluşturulmuş olma ihtimali düşük görünmektedir. Belki de model belli sayıda anahtar noktayı ayırt etme ve çıkarım yapabilme yeteneğine erişmiş olabilir ancak ne eğitimdeki epoch sayısı ne de training verisinin büyüklüğü, bütün anahtar noktaların tam doğrulukla çıkarımını yapabilen ve sınıflandırmalarda böylesine nokta atışı yapan bir modelin ortaya çıkışı için yeterli olmayacaktır.

Her ne kadar problem 7x9 boyutunda çizimlerin yalnızca 7 sınıfa ayrılması gibi temel düzey bir görevi çözmek olsa da, şüphesiz ki böyle bir problemin çözümünü yapacak modelin gelecek yabancı çizimlere hazırlıklı olabilmesi açısından daha çok tecrübelendirilmesi gerekecektir. Ancak yine de eldeki veriler ile en iyi training şartlarının sağlanması için çaba sarfedilmiştir.

Yukarıda bahsedilen nedenler dolayısıyla, bundan sonraki alt başlıklarda yapılacak çıkarımların doğruluğunun sadece mevcut duruma odaklı olduğu ve her zaman gerçek hayattaki problemlere uygun çıkarımlar olmayabileceği, böylesine basit bir yapıdan yapılan çıkarımların karmaşık sinir ağı modelleri için geçerli olmayabileceği not edilmelidir.

4.2 Gözlem

Bu bölümde, yapılan denemelerin sonuçlarının ve bu sonuçlara ulaşma sürecini bize anlatan grafiksel gösterimlerin neleri işaret ettiği irdelenecektir.

4.2.1 Veri Türü

Giriş kısmında belirtildiği üzere, elimizdeki veri Binary kodlanmış ve Bipolar kodlanmış olarak ikiye ayrılmaktadır. Bu veri türünün ikisinin de kullanımı mümkündür ve seçimi programın başında kullanıcı tarafından yapılmaktadır.

Veri türünün karşılaştırılması için en stabil gözlemlene ortamı Tablo 3.3'te gösterilen denemelerde kurulmuştur. Buradan ve buradaki denemelerin Bölüm 3.2'de gösterilen grafiksel gösteriminden yola çıkarak bir karşılaştırma yapılabilir.

Son accuracy değerlerin karşılaştırıldığında, iki modelin de 2000 kez training aşamasından geçmesi sonucunda hemen hemen aynı doğruluk seviyelerine ulaştığı gözlemlenmektedir. Her ne kadar kodlama biçimleri farklı olsa da, bu iki tür de aynı çizimi sembolize ettiği için, yeterince epoch yapıldığı takdirde iki modelin de benzer doğruluğa ulaşacağı görülmektedir.

Bu 4 denemenin grafiksel gösterimi incelendiğinde, modellerin yüksek accuracy değerine yakınsama süreçlerinin çok büyük farklılıklar barındırdığı görülmektedir. Binary veri ile training edilen modelin ilerleyişi zig-zaglı ve tutarsız bir görünümdeyken, bipolar modelin bir değere yakınsama yaparken tutarlı bir şekilde ilerlediği, değişimlerin nadir ve keskin bir biçimde olduğu görülmektedir.

4.2.2 Learning Rule Karşılaştırması

Çalışma kapsamında iki farklı öğrenme kuralının kullanılabildiğinden bahsedilmişti. Bu kurallardan ilki Perceptron ve diğeri Delta öğrenme kuralları olduğu söylenmişti. Bu iki kuralın arasındaki en temel fark, Delta'nın ağırlık güncelleme işleminde referans olarak istenen output ve elde edilen sonucun arasındaki mesafeyi almasıydı.

İki learning rule'un da farklı verilerle eşit sayıda kullanıldığı ve çok sayıda epoch boyunca grafik ilerleyişini izleme şansı bulduğumuz için, learning rule için de Tablo 3.2'deki denemeler referans alınabilir.

Buradaki sonuçlar incelendiğinde, Perceptron ve Delta kurallarının ikisinin de yeterli zaman verildiğinde hemen hemen denk sonuçlar verdiği görülmektedir. Sadece deneme 12'nin sonucunda alınan accuracy değerinin farklı olduğu ve aradaki farkın %2 gibi sözkonusu problem çerçevesinde pek de sorun teşkil etmeyecek bir fark olduğu görülmektedir.

Grafikler incelendiğinde bipolar veri ile yapılan training'in grafiklerinin birbirine benzerlik gösterdiği ancak daha detaylı fikir vermesi açısından binary veri ile yapılan denemelerinin ilerleyişinin daha iyi bir fikir verebileceği görülmektedir. Her ne kadar accuracy değerinde iniş çıkışın fazlalığı ve bu iniş

çıkışların büyüklüğü gibi etmenlerin hepsini detaylı bir şekilde irdelemek zor olsa da, kabaca grafiklerin genel trendi incelenebilir.

Delta kuralı ile eğitime bakıldığında, grafik her ne kadar iniş çıkışlı olsa da iniş çıkışların olduğu her bir alanın medyanı alındığında ortalama accuracy değerinin 1300 epoch'a kadar genel olarak yükselişte olduğu görülmektedir. 1300'den sonra ise iniş çıkış davranışı azalmakta ve accuracy değeri genel olarak korunmaktadır.

Perceptron kuralı ile yapılan eğitim de tıpkı Delta kuralı ile yapılan eğitim gibi çok sayıda iniş çıkış davranışı barındırmaktadır. Fakat bu iki grafikteki iniş çıkışların epoch sayısına göre ilerlemelerinin farklılık gösterdiği görülmektedir. Perceptron kuralı ile eğitilen modelin doğruluğu hızlı bir şekilde 92% değerine atladıktan sonra yaklaşık 200 epoch süresince yaklaşık değerlerde iniş çıkış göstermiş, 250. epoch civarında ise daha yüksek standart bir değer tuturduktan sonra önceki iniş çıkış frekansından daha az bir frekansla bu civardaki değerler arasında zig-zag yapmayı sürdürmüştür. Ancak bu noktadan sonra modelin öğrenmesi ilerlememektedir. Sabit bir range arasında yaptığı iniş çıkışlarının ardından, grafiği sonlarına doğru accuracy değerinden 2%'lik bir kayıp ile ani bir düşüş yaşamış, bu noktadan sonra yaklaşık 500 epoch süresince bu değerde kalarak, iniş çıkış yapmadan eğitimini tamamlamıştır.

4.2.3 Gürültü

Burada gürültü denerek anlatılmak istenen her epoch'ta training verisinde yapılacak rastgele değişimlerdir. Bu değişimlerin miktarı programın başında kullanıcıdan parametre olarak alınmaktadır.

Kabaca düşünüldüğünde, training boyunca verinin tutarlı olması ve sinir ağının bu tutarlı veri üzerinde çalışması daha iyi bir fikir olarak gözükebilir. Ancak burada uygulanan metotla beraber diğer bütün veri modifikasyon metotlarının ortak amacı, modelin yanlış detaylara gereksiz değer yüklemesinin önüne geçerek, rastgele gürültülenmiş verilerde genelde sabit kalan özellikleri hedeflenen değer ile eşleştirmesini mümkün kılmaktır.

Bir örnek üzerinde anlatmak gerekirse, bir sinir ağının obje tanımak gibi bir görevi olduğu varsayalım. Bu sinir ağı çeşitli obje resimleri kullanılarak eğitiliyorsa, sinir ağının bu resimlerde ortak olarak bulunan ama gerçek hayattaki objelerde ortak olarak bulunması şart olmayan bir özelliği ayırt edici olarak algılaması muhtemeldir. Resimlerde ortak olabilen ama gerçek hayatta ortak olmak zorunda olmayan bu özellikler renk, eğiklik, görüntü kalitesi veya başka bir özellik olabilir. Görselin renginde, kalitesinde, temizliğinde veya eğikliğinde yapılacak değişimlerle bu özelliklerin odak noktada olmasının önüne geçilebilir. Böylelikle sinir ağı bir bisiklet gördüğü zaman bunun rengine, eğikliğine, etrafında bulunan ve görüntüyü kısmen bloklayan diğer objelere veya kameranın ilettiği kaliteye değil, bisiklet denince akla gelen genel niteliklere odaklanabilir.

Bu çalışmada ise temel düzeyde rastgele piksel değişikliği yöntemi kullanılmıştır. Çalışmanın başında bunun eğitim sürecine olumlu etki göstereceği öngörülmüştür çünkü yapılacak değişikliklerle, bu çalışmada 14 olarak belirlenen, training verisinin sunduğu olasılıkların artırılması başarılmıştır. Uygulamada bu modifikasyonların etkisini en iyi şekilde gözlemlemek adına Tablo 3.2'deki denemeler ile diğer iki tablodaki denemeler karşılaştırılabilir.

Bu tablolardaki denemelerin en çok dikkat çeken yanları şüphesiz ki ulaştıkları doğruluk seviyeleri ve bu seviyeye ulaşma süreleridir. Özellikle deneme 7, 8, 9, 10, 11'a dikkat edilmelidir. 100% doğruluğa erişen bu modellerin grafiksel gösterimleri incelendiğinde iniş çıkış davranışlarına rağmen genel trend alındığı zaman bütün modellerin yüksek doğruluk seviyesine kararlı ve sürekli olarak yakınsadığı görülmektedir. Başlangıçta özellikle gürültü yoğunluğu en fazla olan 10 ve 11 denemelerinin çok düşük doğruluk seviyelerinden başlayıp 500-600 epoch süresinde tam doğruluğa ulaştığı görülmektedir. Bu kararlılık ve bu doğruluk oranına erişim diğer tablolardaki denemelerde beraber bulunamamaktadır.

4.2.4 Learning Rate

Learning rate bir modelin her adımda ağırlıklarında yapacağı değişim miktarı için bir sınırlayıcı niteliğindedir. Normal şartlarda bir öğrenme sürecini yavaşlatmak mantıksız bir fikir gibi dursa da, özellikle verinin düzenli olarak modifiye edildiği bu çalışma gibi çalışmalarda, fevri artış ve azalışlar bizi istenen ağırlık değerine düzgün bir şekilde yakınsamaktan alıkoyabilir. Bu sebeple çok büyük olmayan bir learning rate ile doğru weight değerine ulaşmak daha iyi bir fikirdir.

Fakat learning rate'in çok küçük seçilmesi de istenen değere çok yavaş ulaşılmasına, hatta durma kriteri epoch sayısı olan traininglerde ulaşılacak maksimum doğruluk oranına ulaşılmasına neden olabilir. Ulaşılacak en iyi doğruluk oranına aşırı yavaş ulaşma durumu bu çalışma kapsamında defalarca gözlemlenmiştir.

Learning Rate'in bu çalışmadaki etkisi en net olarak deneme 4 ve deneme 5 arasında, ayrıca deneme 8 ve deneme 9 arasında gözlemlenebilir. İki karşılaştırmada da, learning rate'i iki katına çıkarmak, sanki learning rate hedefe ulaşma konusunda doğru orantılıymışçasına, hedefe ulaşmak için gereken epoch sayısını yarıya düşürmüştür. Bu problem için learning rate'in aşırı büyük olması her ne kadar grafikte iniş çıkışlara sebep olduysa da, hedefe ulaşma konusunda bir problemle karşılaşılmamıştır.

4.2.5 Aktivasyon Fonksiyonu Eşik Değeri

Aktivasyon fonksiyonu için parametre olan eşik değeri, -1, 0 ve 1 değerlerini almak için aşılması gereken eşiğin sınırlarını belirler.

Bu çalışma kapsamında, özellikle deneme 5 ve 6 arasında da görülebileceği üzere, değişen tek parametre eşik değeri olduğu zaman, erişilecek doğruluk oranı değişmemekte ancak ona ulaşmak için gereken zaman artmaktadır. Aktivasyon fonksiyonunun tanımı ve bu parametrenin görevi düşünüldüğünde beklenen de aslında bu şekildedir.

4.3 Çıkarım

Bu bölümde, yapılan gözlemleri yorumlama yoluyla, hangi değişimin nasıl sonuçlar yarattığı irdelenecektir.

Alt başlık 4.2’de gösterilen ayrı ayrı gözlemlerin sonucunda bu model özelinde bazı çıkarımlar yapmak mümkün olmuştur. Alt başlık 4.1’de belirtildiği üzere, bu çıkarımların bu sinir ağı yapısı, bu problem seviyesi, bu veri seti yapısı ve bu veri büyüklüğü için geçerli olup başka durumlar için geçersiz olma ihtimalinin varlığı göz önünde bulundurulmalıdır. Bunun yanı sıra, sinir ağı modelinin üzerinden kendini eğittiği verinin her epoch’ta belirli oranda modifikasyona uğradığı da hatırlanmalıdır.

Öncelikle verilerin kodlanma biçiminin yarattığı farklılık düşünüldüğünde, tıpkı ilgili bölümde belirtildiği üzere, en temel farklılık eğitim süresince accuracy değerinin gösterdiği değişimlerdir. Her iki kodlama türünün sonucunda da aynı accuracy değerine yakınsandığı düşünüldüğünde, binary şekilde kodlanmış verinin sönük piksellerinin model tarafından “gri” olarak algılanması, bunun yanında aynı durumdaki pikselin -1 ile ifade edildiği için negatif olduğu belli olmasının yarattığı karmaşıklık bu durumun sebebi olarak düşünülmüştür. Perceptron ve Delta kurallarında input layerdaki nodların aldığı değerler formülizasyonda parametre olarak alındığından, bir ağırlığın değişim hızı ve miktarının kodlama ile doğrudan ilişkili olduğu, ancak ulaşılacak accuracy değerinin, yeteri kadar epoch uygulanması şartıyla, iki durumda da ulaşılabilceği görülmektedir.

Başka etkisi gözlenen parametre de öğrenme kuralı olmuştur. Bu iki öğrenme kuralı da benzer accuracy değerlerine yakınsa da, bu yakınsama sürecindeki davranışlarda farklılık gözlemlenmektedir. Bu farklılığın incelenmesi sonucunda, ağırlık güncellemesi için hedeflenen değere olan yakınlığı da göz önüne alarak karar veren Delta öğrenme kuralının daha tutarlı bir yakınsama sağladığı, yine bezner accuracy değeriyle eğitimi tamamlayan Perceptron kuralı için bu sürecin daha iniş çıkışlı ya da başka bir deyişle daha az tutarlı ilerlediği çıkarımı yapılmaktadır.

Bu çalışma boyunca ulaşılan accuracy değerine etkisi en gözle görünür olan parametre gürültü olmuştur. Veride yapılacak rastgele değişim ile gürültü oluşturan ve modelin daha fazla senaryo üzerine eğitilmesi mümkün kılınmıştır. Böylelikle özellikle gürültü düzeyi arttırıldığı zaman, modelin training’de hiç görmediği validation verisini daha iyi yorumlayabildiği, dolayısıyla daha yüksek accuracy değerlerine ulaştığı gözlemlenmiştir. Özellikle eldeki verinin az olması ve verideki piksel sayısının çok olması sebebiyle, overfitting için işlevsel bir çözüm olmuştur. Nitekim gürültünün arttırılmasının hep pozitif sonuçlar verdiği görülmüştür.

Son olarak, training’de öğrenme hızını belirleyen learning rate ve aktivasyon fonksiyonunun parameteresi olan eşik değerlerinin ulaşılacak maksimum accuracy değerine ulaşım hızına etki ettiği ancak bu değer üzerine bir etkisi olmadığı gözlemlenmiştir. Eşik değerinin büyük olması, 1 ve -1 output değerlerine ulaşmak için daha fazla epoch’un gerekmesine sebep olmuştur. Bu problem özelinde learning rate’in yüksek tutulması, verinin küçükülüğünden kaynaklı olarak, olumlu sonuçlar vermiş ve accuracynin hızlı artışını sağlamıştır. Her ne kadar hızlı öğrenme deneme 9’da net olarak görülen kararsız accuracy artışı sonucunu doğursa da, aynı deneme 9 yapılan denemeler arasında 100% validation accuracy değerine en hızlı ulaşan model olmuştur.

4.4 Geliştirilebilirlik

Bu bölümde, oluşturulan yapının ne şekilde ve ne eklemelerle daha komplike gereklilikleri karşılayabileceği konusunda görüşler sunulacaktır.

4.4.1 Problem

Problem kapsamında sağlanan veri ile doğru orantılı olarak, sistemin sadece eldeki çizimleri 7 harfe ayırması beklenmiştir. Ancak daha fazla verinin sağlanması ile beraber daha fazla çıkış varyasyonu sağlanabilir. Programa ayrıca verilen karakter çizimi görselini siyah beyaz hale getirip 7x9 boyutlarına resize eden bir modül implemente edildiği takdirde, programa her türlü görseli işleyip çizilmiş karakteri sınıflandırma yetisi kazandırılabilir. Ancak genişletilmiş problem için yüksek doğruluk oranında sınıflandırmalar yapmak adına sinir ağının katmanlarının arttırılması ve ağın yapısının zenginleştirilmesi konusunda çalışmaların yapılması da gerekebilir.

4.4.2 Veri

Veriler verinin boyutunun yetersizliği ve bu yetersizliğin maskelenmesi için gösterilen uğraşlar hakkında önceki bölümlerde bilgi verilmişti. Bu problem için bile yetersiz kalmış bu verinin zenginleştirilmesi, en azından output nodları aynı kalacak olsa bile farklı fontta verilerin tahsis edilmesi ve verinin 3 sınıfa (training, validation, accuracy) ayrılması, elde edilen modelin kullanılan veriler dışında başka veriler için de doğru tahminler yürütmesi ihtimalini arttıracaktır. Bunun yanında eldeki verinin piksel sayısının arttırılması, program için daha iyi analiz ve daha doğru çıkarımlar yapma şansı doğuracaktır.

4.4.3 Sinir Ağı Yapısı

Her ne kadar şu anki sinir ağı modeli eldeki problem için yeterli olarak gözükse de, eldeki array biçimindeki verinin bir karakter çizimi kodlamasını ifade etmesi, programın amacı ve sınıflandırma şekli göz önünde bulundurulduğunda, Convolutional Neural Network yapısının kullanılması, içeriği genişletilmiş bir problem için daha iyi bir çözüm olacaktır.

4.4.4 Data Augmentation

Veriye eğitim esnasında gürültü uygulanması ve bunun bize sağladığı faydalar önceki bölümlerde irdelenmişti. Ancak 4.4.1’de bahsedilen daha gelişmiş bir problem için bu metodun yeterliliğinin sorgulanması gerekmektedir. 4.2.3’te detaylarıyla anlatılan farklı augmentation metotları uygulanarak, programa daha farklı senaryolar için de doğru tahminler yapma yeteneği kazandırılabilir.