Anıl Özcan 235B7022

Sayısal Video İşleme 2. Ödev Raporu

Özet

Bu çalışma, CIFAR-10 veri kümesi üzerinde nesne tespiti gerçekleştirmek için evrişimli sinir ağı (CNN) kullanmaktadır. CIFAR-10, 32x32 piksel boyutlarında renkli görüntülerden oluşan 10 farklı sınıfa ait bir veri kümesidir. Amacımız, bu görüntülerdeki nesneleri doğru bir şekilde tespit etmek ve sınıflandırmaktır. Bu veri kümesi, gerçek dünya görüntülerine benzer karmaşıklıkta ve çeşitlilikte nesneler içermektedir. Veri seti üzerindeki görseller 32x32 çözünürlükte olmasından kaynaklı insan gözü ile tespit edilmesi zor olsa da nesne tespiti ve derin öğrenme üzerine yapılan geliştirilmelerin pekiştirilmesi adına ilk akla gelen veri setlerinden birisidir.

Bahsedilen nesne tespiti amacının doğrultusunda, öncelikle veri kümesi hakkında detaylı bir bilgi sunulacak ve CIFAR-10'un içerdiği sınıflar ve görüntü özellikleri üzerinde durulacaktır. Daha sonra, kullanılan yöntem ve modelin nasıl uygulandığı açıklanacak ve CNN'nin convoluitonal, pooling ve fully connected layerlardan oluşan yapılarına odaklanılacaktır. Son olarak, eğitim ve test aşamalarının nasıl gerçekleştirildiği ve elde edilen sonuçların değerlendirilmesi sunulacaktır.

Giriş

Proje sürecinde nesne tespiti ve sınıflandırma alanında derin öğrenme yöntemlerini denemek ve gücünü keşfetmek amacıyla CIFAR-10 veri kümesi üzerinde gerçekleştirilen bir projeye dayanmaktadır. Özet kısmında bahsediliği gibi CIFAR-10, 32x32 piksel boyutlarında renkli görüntülerden oluşan ve 10 farklı sınıfa ait geniş bir veri kümesidir. Her sınıf, farklı kategorilere ait nesneler içermektedir; uçak, otomobil, kuş, kedi, geyik, köpek, kurbağa, at, gemi ve kamyon gibi.

Çalışmanın ana hedefi, CIFAR-10 veri kümesi üzerindeki nesneleri doğru bir şekilde tespit edebilen ve sınıflandırabilen bir CNN modeli geliştirmektir. Bu, gerçek dünya uygulamalarında oldukça önemli olan nesne tanıma problemini ele alır. Böylece geliştirilen projenin kazanımları arasında business veya real life problemlere de çözüm getirmek için kullanılabilecek olan CNN modelinin yapısı ve uygulaması süreçleri üzerinde tecrübe kazanmak olarak sayılabilir. Bu kazanım karmaşık ve çeşitli nesneleri içeren CIFAR-10 veri kümesi üzerinde derin öğrenme modellerinin başarısını değerlendirerek sağlanacaktır.

Derin öğrenme, son yıllarda nesne tanıma, görüntü sınıflandırma ve nesne tespiti gibi birçok alanda ağırlığını ispat etmiş ve CIFAR-10 gibi zorlu veri kümeleri üzerinde yapılan çalışmalar, derin öğrenme modellerinin gerçek dünya problemlerinde ne kadar etkili olabileceğini göstermiştir.

Gerçekleştirilen projenin sonucunda nesne tespiti ve sınıflandırma alanında CNN tabanlı modellerin etkinliğini ve CIFAR-10 gibi zorlu veri kümelerindeki performanslarını daha iyi anlamamıza yardımcı olacaktır. Bu nedenle, elde edilen bulgular sonuçlar kısmında paylaşılacaktır.

Yöntem

Projede kullanılan iki farklı CNN modeli, CIFAR-10 veri kümesinde nesne tespiti için özelleştirilmiştir. İlk olarak, üç katmanlı bir CNN modeli (ThreeLayerCNN) tasarlanmıştır. Bu modelde üç konvolüsyon katmanı bulunmaktadır. Her bir konvolüsyon katmanı, 64 adet filtre içermekte olup, her filtre 3x3 boyutundadır. Ardından, ReLU aktivasyon fonksiyonu kullanılarak non-lineer özellikler kazandırılmıştır. Her konvolüsyon katmanının ardından, max pooling işlemi uygulanarak boyut azaltma sağlanmıştır. Overfittingi önlemek ve genelleştirme performansını artırmak amacıyla, dropout yöntemi kullanılarak rastgele bağlantılar geçici olarak devre dışı bırakılmıştır.

İkinci olarak, beş katmanlı bir CNN modeli (FiveLayerCNN) oluşturulmuştur. Bu modelde beş konvolüsyon katmanı bulunmaktadır, her biri 64 adet filtre içermektedir ve her filtre 3x3 boyutundadır. Konvolüsyon katmanlarından sonra ReLU aktivasyon fonksiyonu uygulanmış ve ardından max pooling işlemi gerçekleştirilmiştir. Dropout yöntemi, overfittingi önlemek için her iki modelde de kullanılmıştır.

Her iki modelde de, konvolüsyon katmanlarından sonra tam bağlantılı (fully connected) katmanlar bulunmaktadır. Bu katmanlar, özellik haritalarını düzleştirmek ve sınıflandırma için kullanılmaktadır. Her bir modelin son katmanı, CIFAR-10 veri kümesindeki 10 farklı sınıfa karşılık gelen bir çıkış katmanıdır.

Model mimarileri, derin öğrenme yöntemlerinin nesne tespiti problemlerindeki etkinliğini artırmak amacıyla tasarlanmıştır. Hem üç katmanlı hem de beş katmanlı model, görüntü verilerinden anlamlı özniteliklerin çıkarılmasına ve nesnelerin doğru bir şekilde sınıflandırılmasına odaklanmaktadır. Raporun devamında bu iki model mimarisinin kıyaslanmasına ve loss kıyaslamalarından bahsedilecektir.

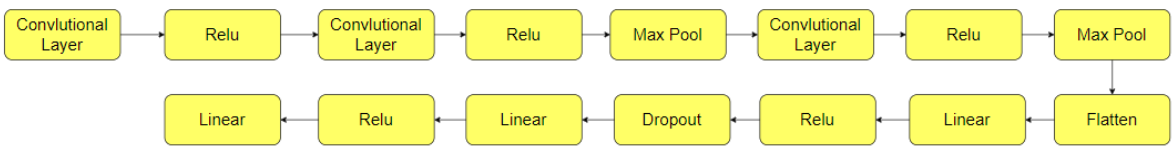

Fig 1. 3-Katmanlı Evrişimli Sinir Ağı Mimarisi

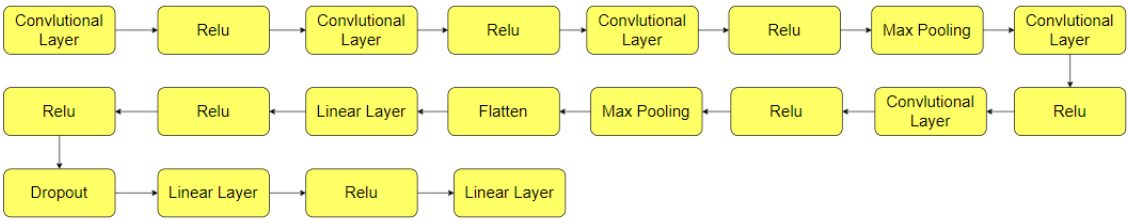

Fig 2. 5-Katmanlı Evrişimli Sinir Ağı Mimarisi

Uygulama

Raporun bu kısmında CIFAR-10 veri setini kullanarak uygulama aşamasından bahsedilecek. İlk olarak, veri setinin internet üzerinden indirilmesi ve ardından bilgisayar ortamına çıkartılması işlemi gerçekleştirildi. Bu adımın temel amacı, modelin eğitimi için gereken veri setini hazırlamaktır. Tabular datanın aksine görseller ile çalışırken daha farklı paketlerden yararlanılır. Bu proje boyunca Google Tarafından geliştirilmiş olan Pytorch kütüphanesinden faydalanılmıştır.

Görseller çalışma ortamına alındıktan sonra, yüklenen veri setinin doğru şekilde yüklendiğinden emin olmak için bazı kontrol işlemleri yapıldı. Bu kontroller, veri setinin içeriğini ve sınıf etiketlerini görmek için yapıldı. Ayrıca, belirli bir sınıfa ait örneklerin sayısını ve örneklerin nasıl göründüğünü kontrol etmek için görselleştirme işlemi yapıldı. Bu adımda yapılan işlemlerin temel amacı, veri setinin doğru şekilde yüklendiğini ve hazır olduğunu doğrulamaktır. Bu, modelin eğitim sürecine geçmeden önce önemli bir adımdır çünkü modelin doğru bir şekilde eğitilmesi için uygun veri setinin hazır olması gerekir. Devamında assignment maddelerinden birisi olan veri setinin train (eğitim), validation (doğrulama) ve test (test) kümelerine bölünmesi ve PyTorch DataLoader objelerine dönüştürülmesi işlemi gerçekleştirilmiştir. Bu işlem, modelin eğitimi için gerekli veri setinin hazırlanmasını sağlamaktadır. Veri setindeki örneklerin %80'i train seti, %20'si ise validation seti olarak ayrılmıştır. Bu ayrım her iki setin de modele doğru şekilde eğitilmesi ve performansının değerlendirilmesi için önemlidir. Oluşturulan train ve validation veri setleri için DataLoader objeleri oluşturulmuştur. DataLoader objeleri, veri setinin batch'leri halinde yüklenmesini sağlamak için kullanılacaktır. Eğitim ve validasyon süreçlerinde bir aksaklık olmaması için loaderların doğru oluşturulması gerekmektedir. Oluşturulan DataLoader objeleri ile bir batch'in içeriği görselleştirilmiştir. Bu sayede, yüklenen verinin doğru şekilde batch'ler halinde ayrıldığını ve modelin eğitimi sırasında kullanılabileceğini kontrol edilmiştir. Ayrıca, görselleştirme işlemi ile veri setindeki örneklerin nasıl göründüğü ve modelin bu örnekleri işleme yeteneği hakkında bir fikir edinilebilir. Loaderlar oluşturulduktan sonra, "ThreeLayerCNN" adında üç katmanlı bir CNN modeli tanımlanmıştır. Bu model, üç evrişimli katman içermekte olup, her bir katmanın ardından ReLU

aktivasyon fonksiyonu ve maksimum havuzlama (maxpooling) işlemi uygulanmıştır. Model ayrıca aşırı uydurmayı engellemek için dropout katmanı içermektedir. Bu modelin amacı, CIFAR-10 veri kümesindeki görüntülerdeki nesneleri tanımak ve sınıflandırmak için kullanılmaktadır. Devamında da "FiveLayerCNN" adında beş katmanlı bir CNN modeli tanımlanmıştır. Bu model, beş evrişimli katman içermekte olup, her bir katmanın ardından ReLU aktivasyon fonksiyonu ve maksimum havuzlama işlemi uygulanmıştır. Ayrıca, aşırı uydurmayı engellemek için dropout katmanı eklenmiştir. Bu iki model mimarisi kullanılarak veri setinin nesne tanıma problemine çözüm getirilmeye çalışılmıştır. Modellerin mimarisi tanımlandıktan sonra eğitim, validasyon süreçlerini uygulamak üzere fonksiyonlar yazılmıştır. Bir derin öğrenme modelinin eğitim süreci özetle: Her bir batch train sürecinden geçer. Train süreci ile kastedilen aslında tahminlerin yapılıp, lossların hesaplanmasıdır. Pytorch tahminleme yapılırken gradyan graflarını oluşturur ve böylece loss olarak hesaplanan tensörlerinin gradyanlarını kullanarak ağırlıkları günceller. Gradyanların hesaplandıktan sonra bu güncelleme Adam optimizer vasıtası ile yapılmıştır. "train_model" fonksiyonu, verilen bir model ve bir batch verisi üzerinde eğitim yapar ve kaybı hesaplar. "validation_model" fonksiyonu ise verilen bir model ve bir doğrulama veri kümesi üzerinde modelin performansını değerlendirir ve kaybı ile birlikte doğruluğu döndürür. Son olarak da, doğruluk hesaplaması için "accuracy" fonksiyonu tanımlanmıştır.

Modellerin ve Tensorlerin GPU'ya Taşınması

Bu aşamada, modellerin ve tensörlerin GPU'ya taşınması işlemi gerçekleştirilmiştir. Bu işlem, model eğitiminde GPU'nun paralel hesaplama yeteneklerinden yararlanarak hesaplama hızını artırmak ve bellek kullanımını optimize etmek için önemlidir. İlk olarak, mevcut cihazı (device) belirleyen get_default_device ve to_device adında iki yardımcı fonksiyon tanımlanmıştır. get_default_device fonksiyonu, mevcut cihazı belirlemek için kullanılırken, to_device fonksiyonu verilen tensörleri belirtilen cihaza taşımak için kullanılır. Bu fonksiyonlar, tensörlerin CPU veya GPU arasında kolayca taşınmasını sağlar. Devamında, veri yükleyicileri (DataLoader) için DeviceDataLoader adında bir sarma sınıfı tanımlanmıştır. Bu sınıf, bir veri yükleyicisini (DataLoader) alır ve her bir veri yığınını belirtilen cihaza taşır. Böylece, model eğitimi sırasında veri yığınlarının doğrudan GPU belleğine yüklenmesi sağlanır. Son olarak, mevcut modellerin ve veri yükleyicilerinin GPU'ya taşınması gerçekleştirilmiştir. to_device fonksiyonu kullanılarak modeller ve veri yükleyicileri (DataLoader) GPU'ya taşınmıştır. Bu işlem, modelin eğitim ve doğrulama süreçlerinde GPU'nun paralel hesaplama yeteneklerinden tam olarak faydalanmasını sağlar. Bu aşamada yapılan işlemler, model eğitimi sürecinin GPU üzerinde daha verimli bir şekilde gerçekleştirilmesini sağlamak için önemlidir. CPU çekirdekleri hızlı

olmasına karşılık GPU üzerindeki çekirdek sayısına göre sayıca az olmalarından kaynaklı proje boyunca biz de GPU desteğinden yararlanıyoruz.

Modeller ve Tensorler GPU'ya taşındıktan sonra kurulmuş model mimarileri 0.001 learning rate kullanılarak eğitime tabi tutuldular. Her bir epoch ile birlikte train ve validation lossları ve validation accuracysi output olarak döndürülmüştür.

```
Model[ThreeLayerCNN] ,Epoch [14], train_loss: 0.1169, val_loss: 1.2914, val_acc: 0.7446
Model[ThreeLayerCNN] ,Epoch [15], train_loss: 0.1198, val_loss: 1.2915, val_acc: 0.7479
Model[ThreeLayerCNN] ,Epoch [16], train_loss: 0.1080, val_loss: 1.2486, val_acc: 0.7456
Model[ThreeLayerCNN] ,Epoch [17], train_loss: 0.1108, val_loss: 1.2805, val_acc: 0.7483
Model[ThreeLayerCNN] ,Epoch [18], train_loss: 0.1054, val_loss: 1.2778, val_acc: 0.7459
Model[ThreeLayerCNN] ,Epoch [19], train_loss: 0.1046, val_loss: 1.3813, val_acc: 0.7427
```

Fig 3. 3-Katmanlı Evrişimli Sinir Ağı Loss ve Accuracy'leri

```
Model[FiveLayerCNN] ,Epoch [15], train_loss: 0.3390, val_loss: 0.8706, val_acc: 0.7377
Model[FiveLayerCNN] ,Epoch [16], train_loss: 0.3147, val_loss: 0.9464, val_acc: 0.7334
Model[FiveLayerCNN] ,Epoch [17], train_loss: 0.2864, val_loss: 0.9836, val_acc: 0.7294
Model[FiveLayerCNN] ,Epoch [18], train_loss: 0.2738, val_loss: 1.0513, val_acc: 0.7316
Model[FiveLayerCNN] ,Epoch [19], train_loss: 0.2457, val_loss: 1.0295, val_acc: 0.7367
```

Fig 4. 3-Katmanlı Evrişimli Sinir Ağı Loss ve Accuracy'leri

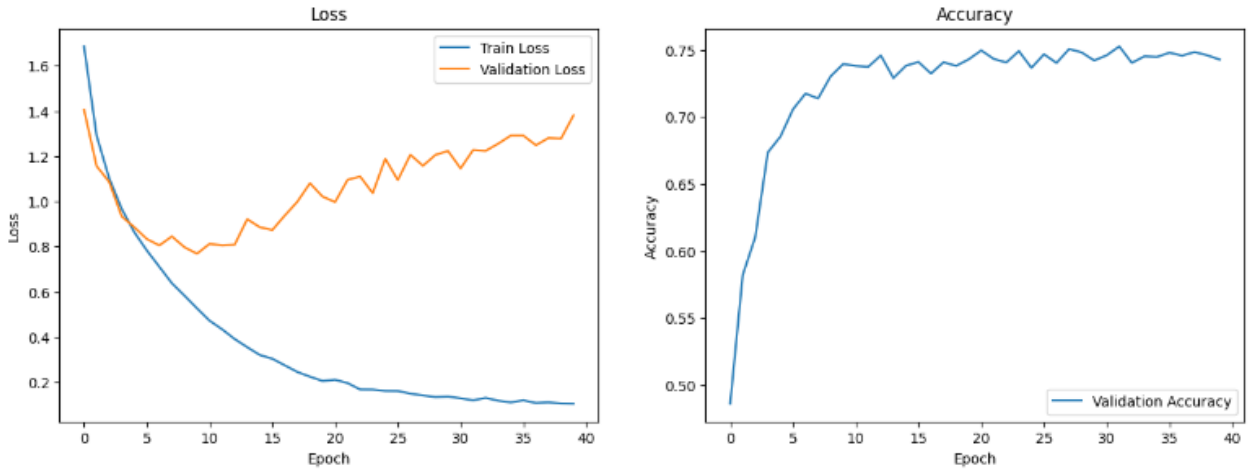

Fig 5. 3-Katmanlı Evrişimli Sinir Ağı Epoch Metrikleri

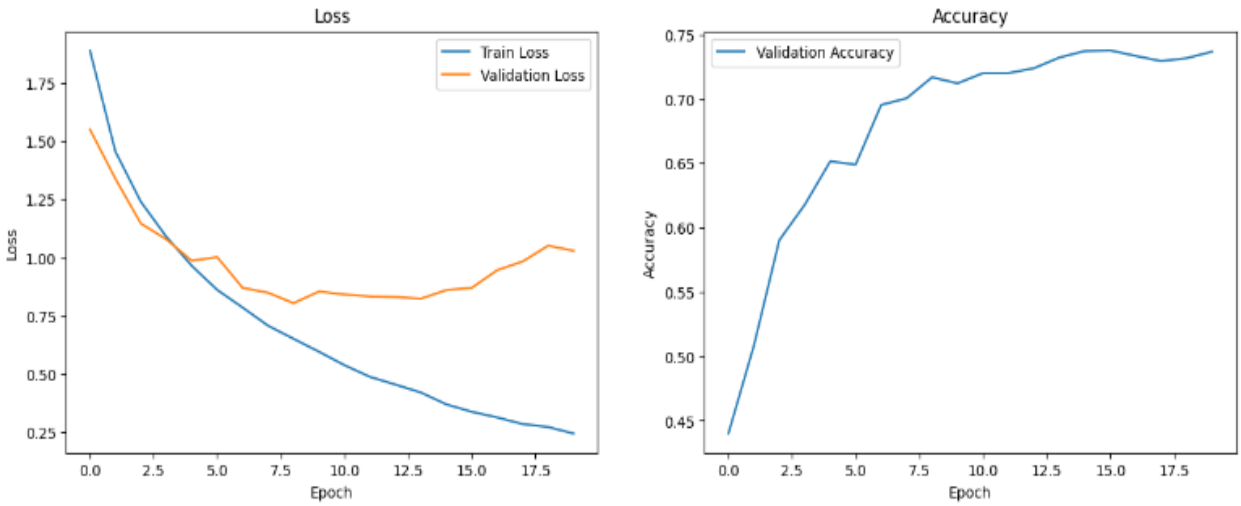

Fig 6. 5-Katmanlı Evrişimli Sinir Ağı Epoch Metrikleri

Epochlar sonrasında 3 katman ile gerçekleştirilmiş CNN modelinin 5 katmanlı Modele göre hafifçe daha başarılı olduğu görülmüştür. Modelin overfit olmaması adına 9 epoch döndürülerek eğitilmesinin en uygun olduğu görülmektedir. Eğitilen modele ait confusion matrix aşağıdaki gibidir.

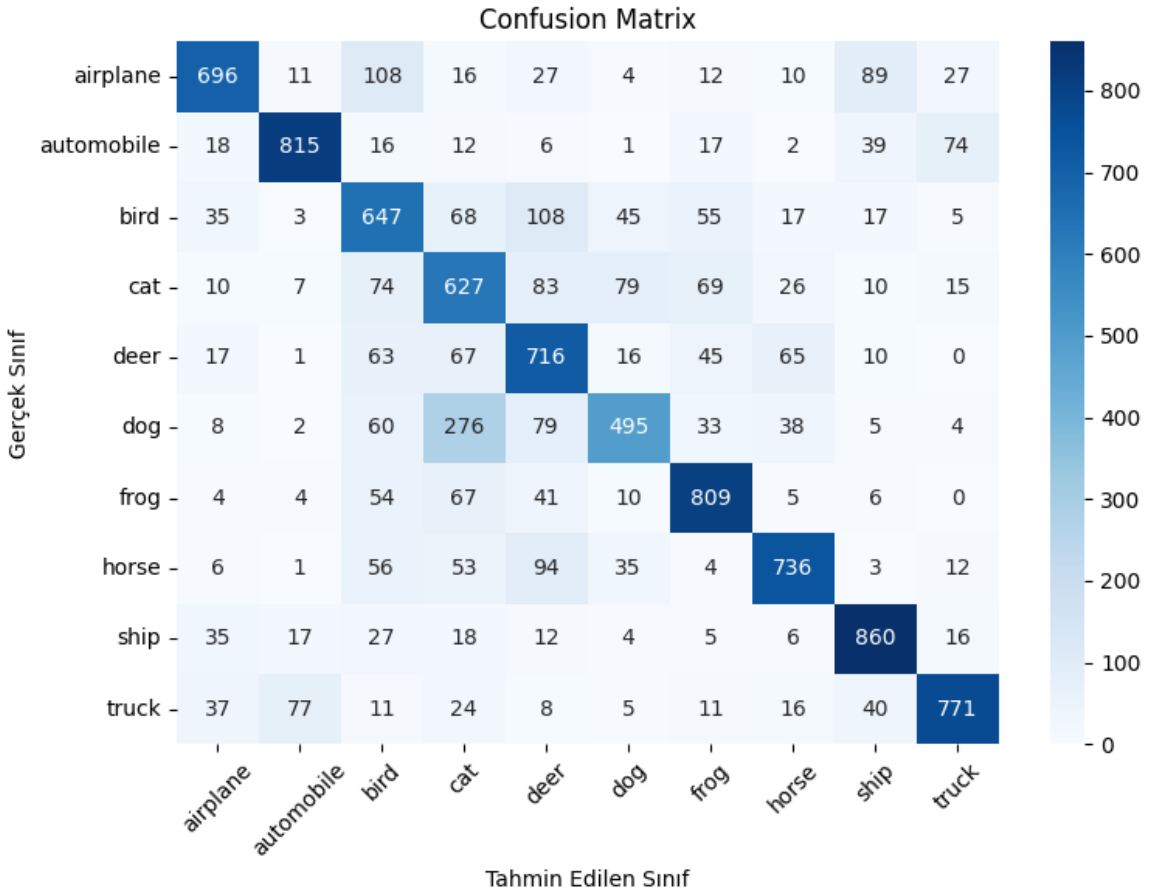

Fig 7. Final Model Confusion Matrix

Başarılı bulunan modelin sınıfların tümümde ezici bir farkla doğru bulduğu Diagonal ısının fazla olmasıyla birlikte de görülebilmektedir. Modelin kedi ve köpek sınıflarının birbirlerine karıştırabildiği görülebilmektedir. Bağlam özelinde 32x32 pikselin yeterli bilgiyi sağlamaması, modelin yeteri kadar komplike olmaması gibi bazı unsurlar buna sebep olmuş olabilir. Bu sınıf dışında tüm sınıflardaki başarı tatmin edicidir. Aşağıdaki görselde de sınıflara ait sample resimlerin en yakın göründüğü 5 resim gösterilmiştir. Confusion matrixe benzer şekilde burada da modelin başarılı olduğu söylenebilir.

```
Class: airplane
- Sample_1: [('airplane', 4.218), ('bird', 3.61), ('cat', -0.591), ('ship', -1.123), ('frog', -1.413)]
- Sample_2: [('airplane', 6.325), ('ship', 2.734), ('truck', 2.428), ('automobile', 0.421), ('bird', -3.279)]
Class: automobile
- Sample_1: [('automobile', 21.121), ('truck', 8.221), ('ship', -2.834), ('airplane', -4.643), ('frog', -10.336)]
- Sample_2: [('automobile', 12.935), ('truck', 4.498), ('ship', -3.011), ('airplane', -4.06), ('bird', -4.425)]
Class: bird
- Sample_1: [('bird', 4.277), ('airplane', 1.719), ('frog', -0.746), ('dog', -0.998), ('cat', -1.982)]
- Sample_2: [('bird', 6.786), ('dog', 1.882), ('cat', 0.798), ('horse', -1.548), ('airplane', -1.675)]
Class: cat
- Sample_1: [('cat', 6.513), ('dog', 5.531), ('frog', 1.203), ('bird', -0.4), ('horse', -3.728)]
- Sample_2: [('cat', 3.524), ('bird', 2.335), ('dog', 2.009), ('frog', 1.996), ('deer', 1.537)]
Class: deer
- Sample_1: [('deer', 7.574), ('dog', 0.556), ('bird', 0.537), ('horse', 0.32), ('cat', 0.01)]
- Sample_2: [('deer', 7.475), ('horse', 2.138), ('dog', 0.658), ('bird', 0.428), ('cat', -0.139)]
Class: dog
- Sample_1: [('cat', 3.335), ('dog', 2.75), ('bird', 1.771), ('frog', -0.034), ('horse', -1.675)]
- Sample_2: [('cat', 4.42), ('dog', 2.966), ('bird', 1.832), ('airplane', 0.471), ('frog', -1.682)]
Class: frog
- Sample_1: [('frog', 8.466), ('cat', 3.523), ('dog', -0.155), ('bird', -1.997), ('truck', -2.253)]
- Sample_2: [('frog', 6.899), ('bird', 1.52), ('cat', 0.684), ('deer', -0.333), ('dog', -0.835)]
Class: horse
- Sample_1: [('horse', 14.73), ('deer', 2.513), ('dog', 1.654), ('bird', -2.198), ('cat', -5.903)]
- Sample_2: [('horse', 5.178), ('dog', 0.624), ('deer', -1.113), ('bird', -1.358), ('airplane', -2.351)]
Class: ship
- Sample_1: [('ship', 6.785), ('airplane', 5.955), ('truck', 2.001), ('automobile', 0.221), ('bird', -4.58)]
- Sample_2: [('ship', 9.252), ('automobile', 4.552), ('airplane', -0.439), ('truck', -2.56), ('cat', -3.576)]
Class: truck
- Sample_1: [('truck', 10.54), ('automobile', 3.581), ('airplane', -2.405), ('cat', -4.439), ('ship', -4.493)]
- Sample_2: [('truck', 5.852), ('horse', 0.084), ('airplane', 0.049), ('dog', -2.616), ('cat', -3.262)]
```

Fig 8. Sample Resimlerin Tahmin Sınıfları

Sonuç

CNN ile nesne sınıflandırma, doğru hiperparametrelerin seçimi ve uygun model mimarilerinin kullanımıyla etkili bir şekilde gerçekleştirilebilir. Bu çalışma, belirli bir veri kümesi üzerinde yapılan deneylere dayanarak, üç katmanlı CNN modelinin beş katmanlı modele göre daha iyi performans göstermiştir. Test setinde elde edilen 0.73 başarı CNN modellerinin obje tanıma üzerinde başarılı olabileceğine ışık tutmaktadır.