KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROJENİN KONUSU

ANOMALİ DURUMLARIN TESPİTİ

BİTİRME PROJESİ

Adı SOYADI

330104 ONUR ERDAŞ

340824 EMRE ÖZDEMİR

2019-2020 BAHAR DÖNEMİ

KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROJENİN KONUSU

ANOMALİ DURUMLARIN TESPİTİ

BİTİRME PROJESİ

Adı SOYADI

330104 ONUR ERDAŞ

340824 EMRE ÖZDEMİR

2019-2020 BAHAR DÖNEMİ

IEEE Etik Kuralları

IEEE Code of Ethics

Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayırımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da onaylanmıştır.

ÖNSÖZ

Projeye başlarken bir merak üzerine girdiğimiz ve bilmediğimiz bir alan olan makine öğrenmesi konusunda çalıştık. Bu çalışma birçok şeyi anlamamıza ve daha iyi yorumlamamıza fayda sağladı. Bu açıdan Yapay Sinir Ağları'na girmemiz kendi açımızdan çok verimli olduğu kanısındayız. Proje ile birlikte makine öğrenmesinin genel mantığını ve bazı kütüphanelerinin C-C++ ile nasıl kodlanabileceğini öğrendik. Çeşitli denemeler sonucu ile videodaki anomali durumların tespitini elde etmeye çalıştık. Bu denemeler ile ağ yapısı üzerinde çalışmayı öğrendik ve çeşitli değişikliklerin performansı nasıl artırabileceğini gözlemledik.

Bu projede öncelikle desteklerini esirgemeyen kıymetli ailemize, emeği geçen ve her zaman bize yol gösteren Sayın Hocamız Prof. Dr. Murat EKİNCİ'ye ve değerli arkadaşlarımıza teşekkürlerimizi sunuyoruz.

ONUR ERDAŞ
EMRE ÖZDEMİR
Trabzon 2020

İÇİNDEKİLER

ÖZET

Projenin amacı bir video akışı esnasında o an herhangi bir anormal olayın gerçekleşip gerçekleşmediğinin tespit edilmesidir. Bu amaçla çıkılan yolda Yapay Sinir ağlarıyla ilgili geniş çaplı bir çalışma gerçekleştirilmiştir. Tespit edilmesi hedeflenen anormal durumlar belirlenmiştir. Bu belirlenen anormal durumlardan sonra CNN ortamında videolarda nasıl bir çalışma gerçekleştirileceği üzerine çeşitli araştırmalar ve hedeflenen noktaya nasıl gelineceği üzerine çalışmalar gerçekleştirilmiştir. Belirlenen başlıklarda anormal duruma ait videoların frame'ler şeklinde ayrılarak kendi isimlerinde belirli bir mantıkla kaydedilme işlemi gerçekleştirilmiştir. Bu frame'leme işleminden sonra videolardan çıkarılan bu frame'leri ağa etiketleyerek verilme işlemi yapılmıştır. Bu şekilde hangi frame topluluğunun hangi olaya ait olduğu veya herhangi bir olay olmadığı ağa verilen bu etiketler sayesinde öğretilmiştir. Bu işlemler ardından ağda çeşitli denemeler ve iyileştirme denemeleri yapılmıştır. Bu çalışmalardan sonra optimum ağ yapısına karar verilmiştir. En iyi performans sergileyen ağ seçildikten sonra ağa verilen frame'lerin dağılımları eşitlenmiş ve sonuçlar gözlemlenmiştir. Örnek ağ yapıları C++ ile kodlanmış ve işlemlerin nasıl gerçekleştiği öğrenilmiştir. Proje bu şekilde gerçekleştirilmiştir. Projenin bir sonraki adımında gömülü sistem ile bir kameraya entegre edilip denemeler yapılabilir. Sonuçların daha iyi gözlemlenmesi için dataset genişletilmelidir. Yazılan bu tez bir sonraki kişilere kaynak olup projeyi daha ileri taşımaları hedeflenmiştir.

1. GENEL BİLGİLER

1.1. Giriş

1.1.1. Problem

Projenin amacı, verilen videoda yangın, patlama, hırsızlık v.b gibi anomali bir durumla karşılaşıldığında bunun tespit edilmesi şeklinde özetlenebilir. Bu proje mobese kameralarına uyarlandığında, herhangi bir yangın veya patlama durumunda acil müdahale birimlerine veri akışı sağlanarak olay yerine acil şekilde intikal sağlanabilir. Bu durumda herhangi bir bireyin acil müdahale birimlerine haber vermesi gerekliliği ortadan kalkar. Bu şekilde can ve mal kaybının en aza indirilmesi sağlanabilir.

Elimizde bulunan veri kümesinde etiketlemeleri doğru ve hassas şekilde yapılması ile olayın doğru algılanması sağlanacaktır. Anomali ve normal durumların etiketlenmesi sonrasında bu verilerin kurulan ağa verilerek, bu durumların eğitim ile ağa öğretilip ağdan doğruya yakın bir sonuç alınması beklenmektedir.

1.1.2. Yapılan Çalışmalara Giriş

Tasarım aşamasında yapılan işlere ek olarak anomali durum sayısı genişletilmiş ve bu işlemle birlikte kullanılan dataset büyütülmüştür. Daha büyük bir dataset üzerinde çalışılmıştır. Dataset'in büyük olması programın iyi çalışacağı anlamına gelmemektedir. Ağa verilen frame paketlerinin dağılımları düzenlenerek daha iyi bir sonuç elde edilmiştir.

Tasarım aşamasında kullanılan ağ yapısı yeterli görülmeyerek birçok deneme ile testler yapılmış ve sonuçları kayıt edilmiştir. Yapılan testler sonucunda özgün ağlar kurulmuş ve programın çalışması özgün ağ ile devam ettirilmiştir. Ayrıca test aşamasında denenen ağlardan bazıları günümüzde oldukça iyi bilinen VGG16 ve ResNet ağ denemeleri gerçekleştirilmiştir. Bu denemelere ek olarak Pooling yokluğu, aktivasyon fonksiyonu değişikliği, Convolution sayılarının ve parametrelerinin değiştirilmesi ile denemeler gerçekleştirilmiştir. Bu denemeler sonucunda en iyi *accuracy* ve en düşük *loss* değerlerine sahip olan ağ yapısı seçilmiştir.

Yapay sinir ağlarında yaptığımız tüm bu olayların arkasında nasıl bir kodlama yapıldığının anlaşılması ve bakış açısını genişletmesi açısından bu fonksiyonların C++ kısmında yazılması da hedefler arasındadır. Bu hedefle yola çıkarak ilk olarak Full Connected bir ağ

yapısını Python'da eğitip ağırlık katsayılarını alıp C++'da bu ağırlıklar ile Python'da kullanılan ağ yapısının kodlanması gerçekleştirilmiştir. Sonuçlar karşılaştırılarak aynı sonuçlar elde edilmiştir. Bu kısımdan sonra CNN yapısına geçiş yapılmıştır. İlk olarak Conv2D üzerinde Cifar10 dataset'inden faydalanarak bir eğitim gerçekleştirilmiştir. Bu eğitim sonucunda elde edilen ağırlıklar bir txt dosyasına kayıt edilip C++ kısmında kullanılmıştır. C++'ta bu ağırlıklar ile Python'da kurulan CNN ağ yapısı kodlanmıştır ve test işlemi gerçekleştirilmiştir. Elde edilen sonuçlar Python'dan elde edilen sonuçlarla karşılaştırılmıştır ve bu karşılaştırmalar sonucunda aynı sonuçlar elde edilmiştir. Son olarak Conv3D ile çalışılmıştır. Burada ise Python'da basit bir ağ yapısı kurularak basit şekilde eğitim yapılmıştır. Bu eğitim sonucunda elde edilen ağırlıklar txt dosyasına aktarılmıştır. C++ kısmında bu ağırlıklar okunarak Python kısmında kurulan ağ yapısı kurulmuştur ve test işlemi gerçekleştirilmiştir. Python'dan elde edilen ara katman ve sonuç değerleri ile C++'ta elde edilen ara katman ve sonuç değerleri karşılaştırılmıştır. Bu karşılaştırmalar sonucunda aynı sonuçlar elde edildiği gözlemlenmiştir.

1.1.3. Elde Edilen Sonuçlar

Hedeflenen amaçlar başarıyla yerine getirilmiş ve projenin çalışması %70 üzerinde bir doğrulukla gerçekleştirilmiştir. Problemlerin çözümü yapılan çalışmalar kısmında verilmektedir.

2. YAPILAN ÇALIŞMALAR

2.1. Dataset Oluşturulması

Projede ilk aşama belirli bir dataset'in toplanması ve etiketlenmesinden oluşmaktadır. Sayın Hocamız Prof. Dr. Murat EKİNCİ tarafından aldığımız dataset linki ile gerekli dataset indirilmiştir. İndirme sonrasında dataset içerindeki videolar frame'lere ayrılmış ve etiketlemeleri yapılmıştır. Gerekli kodların yazılması sonrasında küçük bir dataset ile gerekli çalışmalar yapılmıştır. Yapılan çalışmalarda video sayıları eşit sayıda verilirken her olaydan kaç paket frame'in verildiği ve olaylar için frame dağılımları göz önüne alınmamıştır. Ağa verilen herhangi bir olayın fazla olması durumunda, ağın eğitimi yapıldıktan sonra, video ile ağın testi yapılırken fazla dağılımı olan olayın en fazla görüntülendiği tespit edilmiştir. Bu nedenle öncelikle ağa verilen frame dağılımların eşitlenmesi ve sonrasında ağa verilen paketlerin sayılarının eşitlenmesi işlemleri yapılmıştır. Yapılan işlemlere göre çıkış grafikleri çizdirilmiş ve yorumlanmıştır. 2.3 başlığı altında bu grafikler ve yorumlar bulunmaktadır.

İlk denemede eğitim için eşit sayıda frame paketleri ağa verilmemiştir. Ayrıca ağa verilen frame dağılımları eşitlenmemiştir. Ağa verilen frame paketlerinin sayısı aşağıdaki gibidir.

	Anormal Durum 20'lik Frame Paketi Sayısı	Normal Durum 20'lik Frame Paketi Sayısı
Explosion	121	475
Fighting	1037	991
Burglary	552	711
Vandalism	221	484
Arrest	717	748
Normal Video	0	875

Tablo1- İlk denemede ağa verilen paket sayıları

İkinci denemede eğitim için eşit sayıda frame paketleri ağa verilmemiştir. Yalnızca ağa verilen frame dağılımları her olay için eşit verilmeye çalışılmıştır. Ağa verilen frame paketlerinin sayısı aşağıdaki gibidir. Bu aşama ile videolardaki olayların tespit durumu çok daha iyi çalışmaktadır.

	Anormal Durum 20'lik Frame Paketi Sayısı	Normal Durum 20'lik Frame Paketi Sayısı
Explosion	630	681
Fighting	384	761
Burglary	500	658
Vandalism	470	754
Arrest	589	675

Tablo2- Ağa verilen frame dağılımlarının eşitlenmesi sonucunda oluşan paket sayıları

Üçüncü denemede ise eğitim için eşit sayıda frame paketleri ağa verilmiştir. Ağa verilen frame paketlerinin sayısı aşağıdaki gibidir.

	Anormal Durum 20'lik Frame Paketi Sayısı	Normal Durum 20'lik Frame Paketi Sayısı
Explosion	600	0
Fighting	600	0
Burglary	600	600
Vandalism	600	0
Arrest	600	0

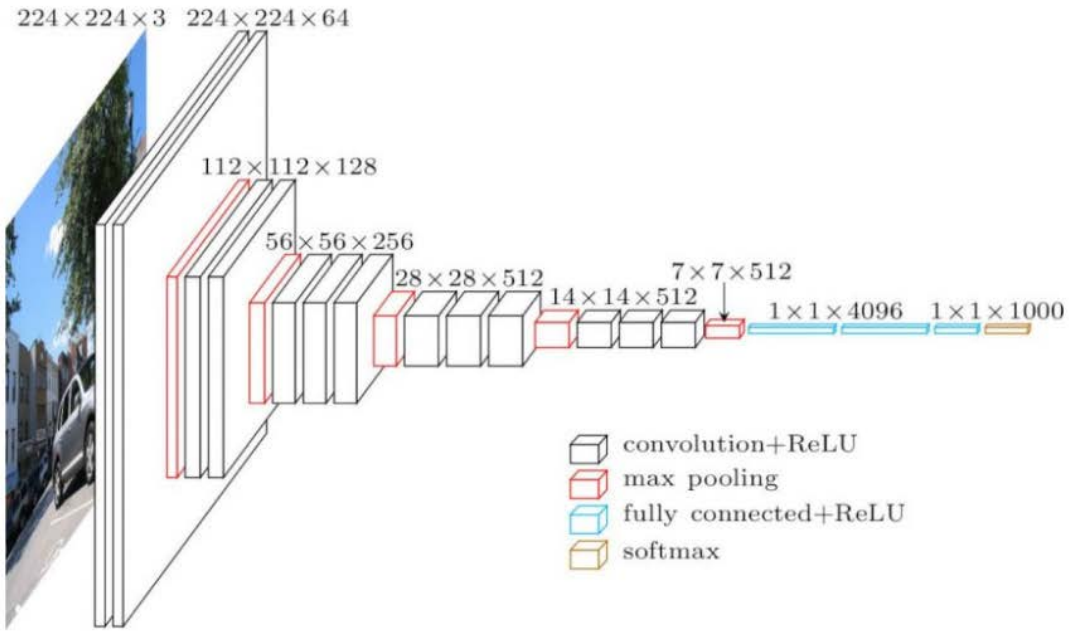
Tablo3- Sınıflarda bulunan frame paketlerinin eşitlenmesi

Ağa verilen frame paketlerinin sayılarının eşitlenmesi ile ağın bir olaya karar verme yeteneği geliştirildi. Ağdaki normal olayların azaltılması ile dataset küçülmüştür ancak videodaki olayların tespiti daha iyi yapılmaktadır. Dataset eşit sayıda frame paketleri ile büyütüldüğünde videodaki olayların tespiti çok daha iyi yapılacaktır.

2.2. Ağ Denemeleri

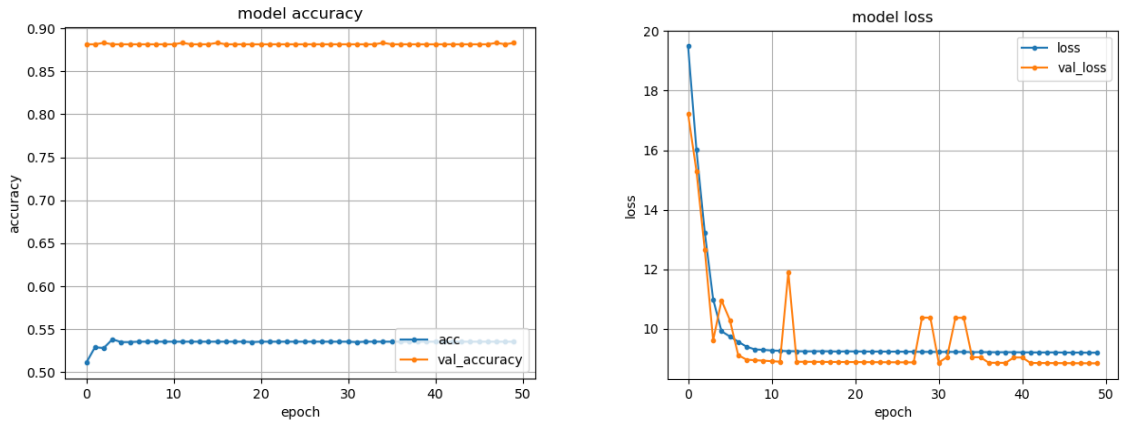
2.2.1. VGG16 Ağı

VGG16 ağı bir Convolutional Neural Network modelidir. K. Simonyan ve A. Zisserman tarafından ortaya atılmıştır. Bu ağ daha önce denenmiş ve üzerinde çalışılmış bir ağ modelidir. Bu yüzden geliştiriciler kendileri bir ağ tasarlamak yerine bu ve bunun gibi ağları kullanmaktadır. VGG16 ağ modeli aşağıdaki gibidir:



Şekil-1 VGG16 Mimarisi

VGG16 modelinin ağa kullanılması sonucunda ortaya çıkan grafikler aşağıdaki gibidir:

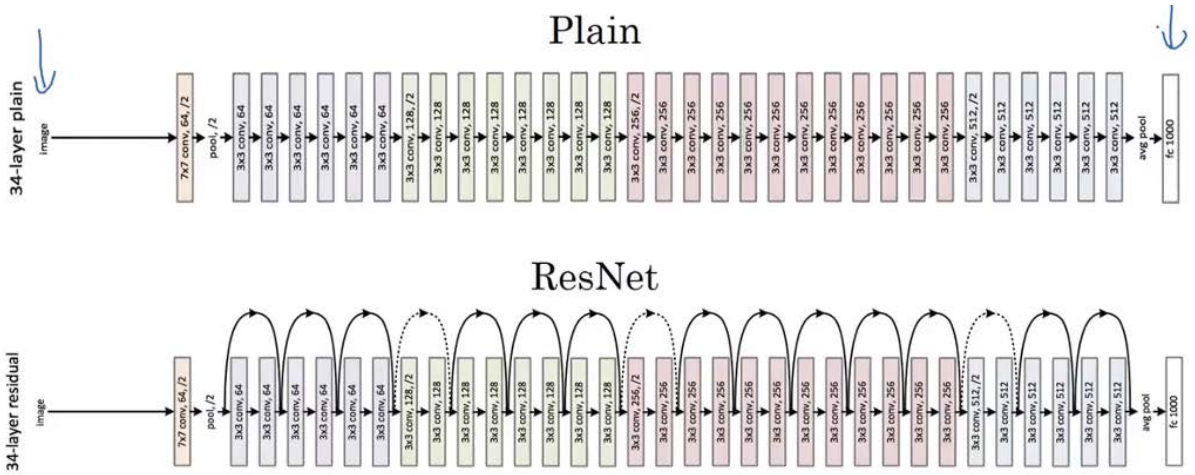


Şekil2- VGG16 Ağ mimarisinin kullanılması sonucu elde edilen sonuçlar

Yukarıdaki şekilde eğitim için accuracy ve loss grafiklerini inceleyecek olursak accuracy %55 altında kalmış ve iyi bir sonuç vermemektedir. Benzer şekilde loss değeri yaklaşık 8 değerinde kalmıştır. Loss değeri 0'ın altında kalmalı ve accuracy değeri %70'in üzerinde olmalıdır. Test için accuracy ve loss değerleri, eğitim için olan accuracy ve loss değerleri kadar önemli değildir. Ancak eğitim ve test için accuracy değeri %100'e yakın olmalı ve loss değeri 0'a yakın olmalıdır. Bu nedenle bu ağ modeli projedeki kullanılan ağ için seçilmemiştir.

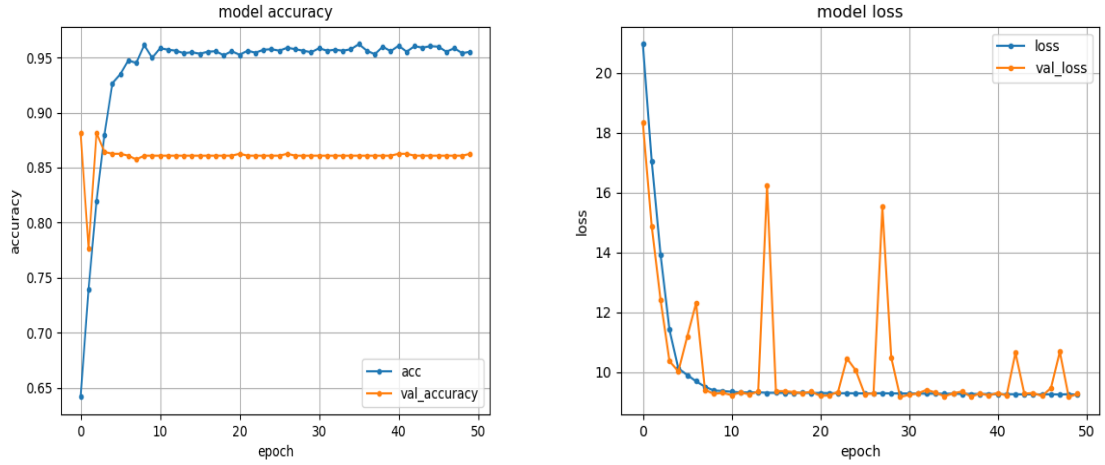
2.2.2. ResNet

ResNet de hazır kullanılan bir CNN modelidir. Bu ağ yapısının mimarisi aşağıdaki gibidir:



Şekil3- ResNet Mimarisi

ResNet modelinin ağa kullanılması sonucunda ortaya çıkan grafikler aşağıdaki gibidir:



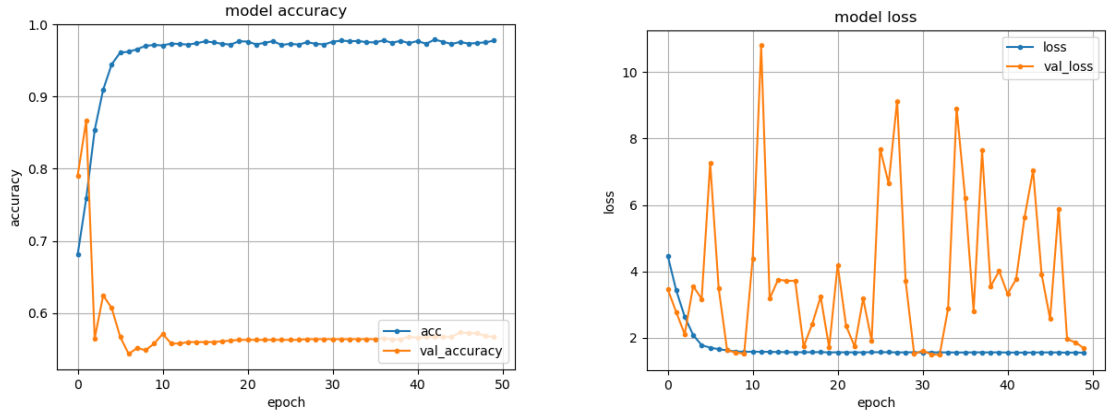
Şekil4- ResNet mimarisinin uygulanması sonucu elde edilen sonuçlar

Yukarıdaki şekilde ResNet mimarisi için accuracy ve loss fonksiyonlarının grafikleri görülmektedir. Eğitim için accuracy değeri oldukça iyi bir değerde olmakla birlikte loss değeri 0'a çok yaklaşmamıştır. Projede aynı dataset ile ResNet mimarisi ile eğitim yapılmış ve sonuçlar gözlemlenmiştir. Gözlemlenen bu sonuçlara göre ResNet mimarisi ile eğitim yapılmış modelin çıktıları çok iyi sonuç vermemiştir. İyi sonuç alınmamasının sebebi loss değerinden dolayı olduğu düşünülmektedir. Bu nedenle projedeki ağ yapısında ResNet mimarisi kullanılmamıştır.

2.2.3. Frame Sayısındaki Değişiklikler

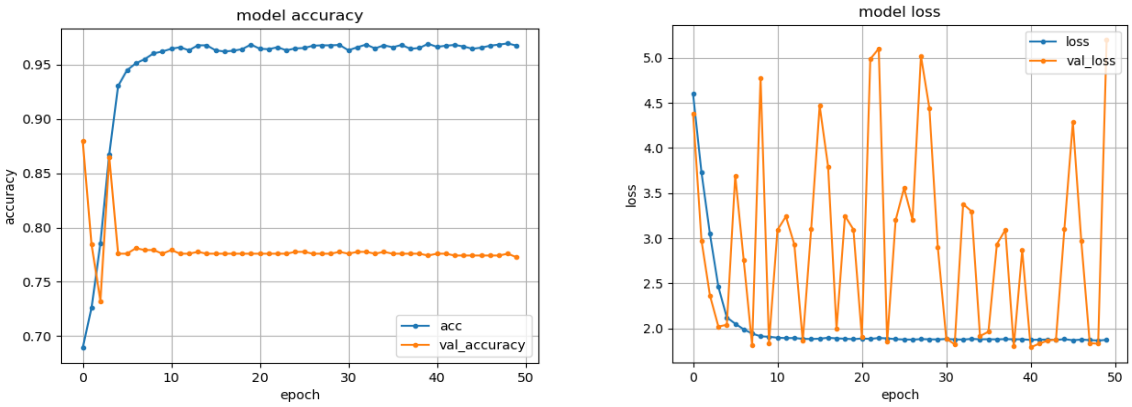
Kullanılan ağlardan en iyi hangisinden verim alınacağının araştırılmasının yanı sıra ağa verilen frame sayısında da değişiklikler yaparak ağın bu değişikliklere nasıl sonuç verildiği de irdelenmiştir. Tasarım aşamasında ağa tek seferde 16 frame gönderilmekteydi. Bu sefer ise ağa tek seferde 12 frame ve 20 frame gönderilerek verdiği sonuçlar irdelenmiştir. Çizdirilen grafiklerde accuracy ve loss değerlerine bağlı olarak seçim yapılması gerekmektedir. Grafiklerin incelenmesi sonucunda ve Sayın Hocamız Prof. Dr. Murat EKİNCİ ile kararlaştırılarak ağa gönderilen bir paketteki frame sayısının 20 frame olması belirlenmiştir.

Ağa gönderilen frame sayısı 12 ve 12 frame içinde 9 tanesinin etiketinde olay olduğunda elde edilen sonuçlar:



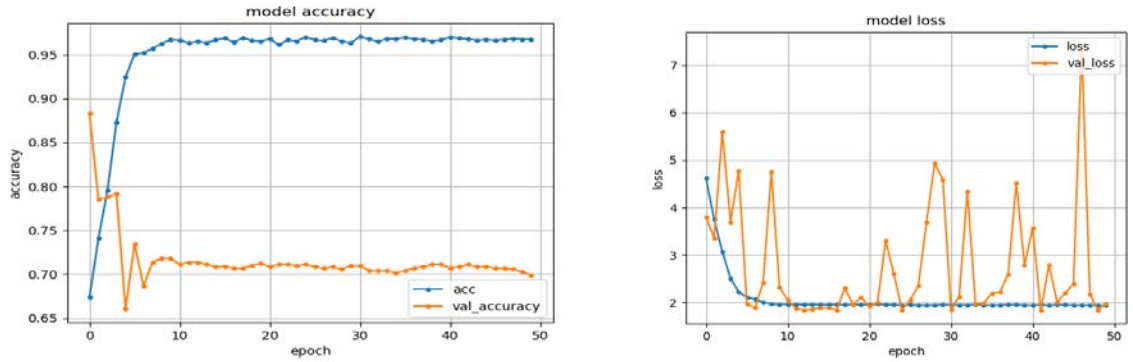
Şekil5- Ağa 12 frame gönderimi sonucunda elde edilen sonuçlar

Ağa gönderilen frame sayısı 16 ve 16 frame içinde 8 tanesinin etiketinde olay olduğunda elde edilen sonuçlar:



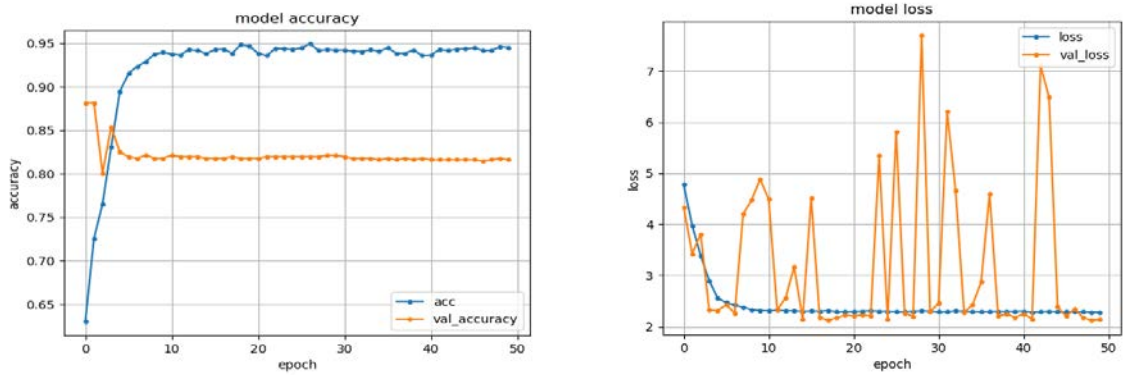
Şekil6- Ağa gönderilen 16 frame'den 8'inde olay olması sonucu

Ağa gönderilen frame sayısı 16 ve 16 frame içinde 12 tanesinin etiketinde olay olduğunda elde edilen sonuçlar:



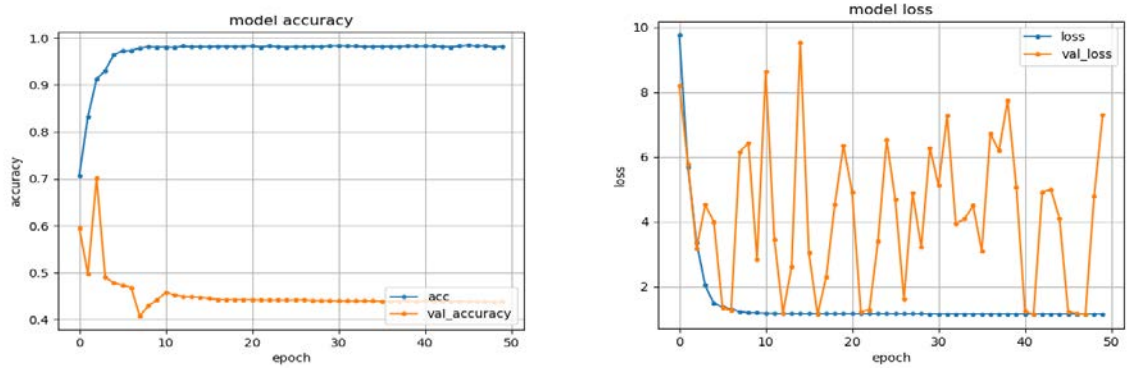
Şekil7- Ağa 16 frame gönderimi sonucunda elde edilen sonuçlar

Ağa gönderilen frame sayısı 20 ve 20 frame içinde 15 tanesinin etiketinde olay olduğunda elde edilen sonuçlar:



Şekil8- Ağa 20 frame gönderimi sonucunda elde edilen sonuçlar

Ağa gönderilen frame'lerde her saniyede birer frame atlayarak gönderilmesi sonucunda elde edilen sonuçlar:

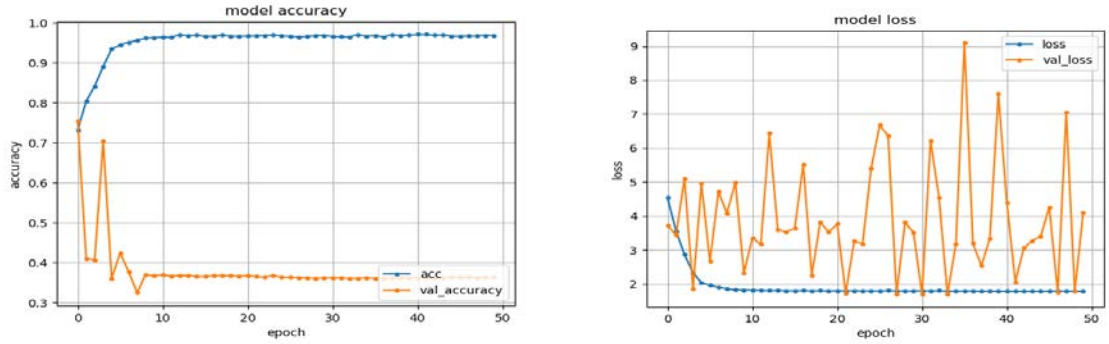


Şekil9- Her saniyede 1 frame atlama sonrası elde edilen sonuçlar

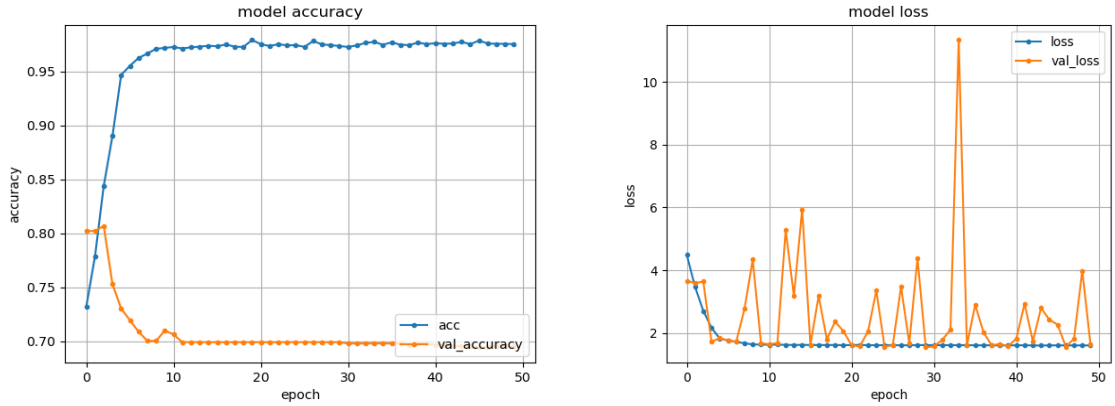
Yukarıdaki grafikler incelenecek olunursa en iyi accuracy ve loss değerlerinin ağa gönderilen frame paketleri için 20 frame sayısının uygun olduğu gözlemlenebilir. Ağa gönderilen 20'lik frame grubu için 15 tanesinde olay olması durumunda olay var denmiştir. Yukarıdaki deneme sonuçlarına daha fazla deneme eklenilebilir ve daha iyi sonuç veren frame sayısı bulunabilir.

2.2.4. Test ve Train Dataset'lerdeki Değişiklikler

Tasarım aşamasında test için 1 video, train için 4 video dataset'i oluşturularak çalışılmaktaydı. Bitirme aşamasında ise test ve train kısmında da değişiklikler yaparak ne gibi değişiklikler olduğunu gözlemlemeye çalışıldı. Bu sebeple çeşitli denemeler yapıldı. Bu denemeler sonucu elde edilen sonuç grafikleri aşağıdaki gibidir:



Şekil10- 3 adet train 2 adet test verisi için sonuç grafikleri

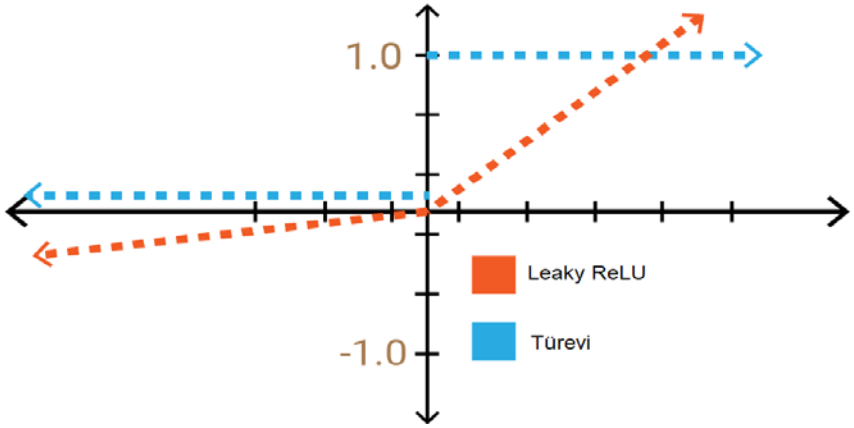


Şekil11- 4 adet train 1 adet verisi test için sonuç grafikleri

Benzer şekilde yukarıdaki grafikler incelendiğinde 4 eğitim ve 1 test videosu ile oluşturulmuş dataset kullanıldığında elde edilen accuracy ve loss değerleri daha iyi sonuçlar vermektedir. Dataset'in %80'i eğitim için %20'si test için ayrılmaktadır.

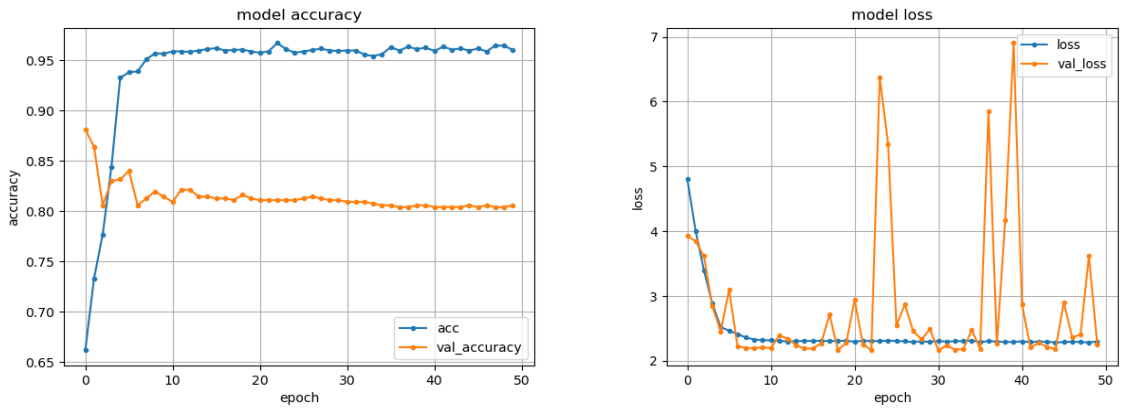
2.2.5. Leaky ReLU Kullanımı

Çeşitli denemeler yapılırken aktivasyon fonksiyonunda bir değişiklik yapıldığında nasıl bir değişiklik olacağı merak konusu olunca Leaky ReLU aktivasyon fonksiyonu kullanılarak sonuçlar gözlemlenmek istenmiştir. Diğer ağlarda aktivasyon fonksiyonu olarak Relu fonksiyonu kullanılmıştır. İki aktivasyon fonksiyonun arasındaki farkları gözlemlemek için aynı ağ yapısı için yalnızca aktivasyon fonksiyonu değiştirilerek sonuçlar gözlemlenmiştir. Leaky ReLU ile eğitilen ağın accuracy ve loss grafikleri aşağıda gözlemlenmektedir. Relu aktivasyon fonksiyonu ile karşılaştırıldığında accuracy ve loss değerleri daha düşük alınmaktadır. Bu nedenle projede aktivasyon fonksiyonu olarak Relu fonksiyonu kullanılmıştır.



Şekil12- Leaky ReLU Aktivasyon Fonksiyonu

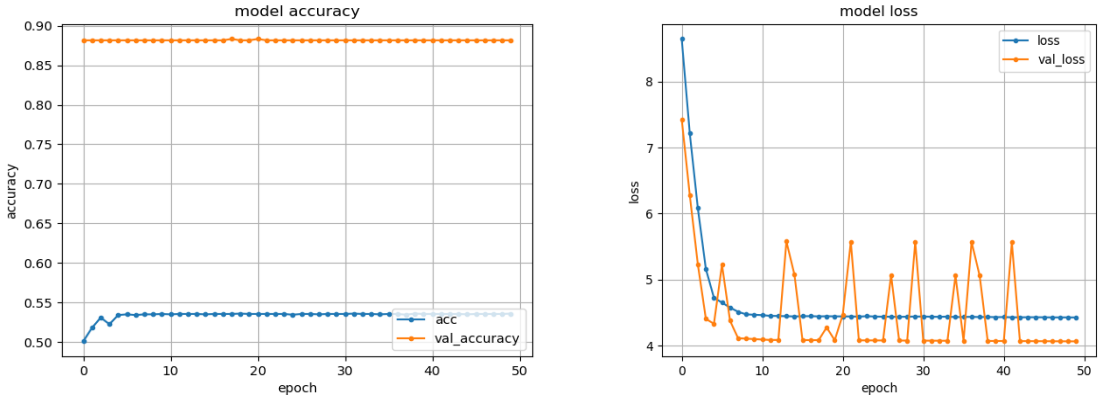
Leaky ReLU kullanımı sonucunda elde edilen sonuçlar aşağıdaki gibidir:



Şekil13- Leaky ReLU sonucunda elde edilen sonuçlar

2.2.6. Pooling Kaldırılması

Kurulan CNN ağı içerisinde MaxPooling kullanılmaktadır. Deneme olarak bu aşamada MaxPooling katmanları kaldırılıp frame'lerdeki boyutları Convolution katmanı içerisindeki *strides*'lar değiştirilerek ağ denemesi yapılmıştır. Convolution içerisindeki strides'lerin değiştirilmesi aslında MaxPooling katmanı ile yapılan boyut indirgeme işlemi gerçekleştirilmiştir. Ancak kernel matrisi ile frame üzerinde gezinirken maksimum piksel seçme işlemi yapılmamıştır. Aşağıda strides değiştirerek elde edilen ağ yapısının sonuçlarının grafikleri gözlemlenmektedir. Bu ağın sonuçları yorumlandığında eğitim için accuracy değeri düşük çıkmaktadır. İstenilen şekilde performans vermemiştir. Bu yüzden bu ağ modeli kullanılmamıştır. Bunun yerine projede seçilen ağ modelinde MaxPooling katmanı kullanılmıştır.



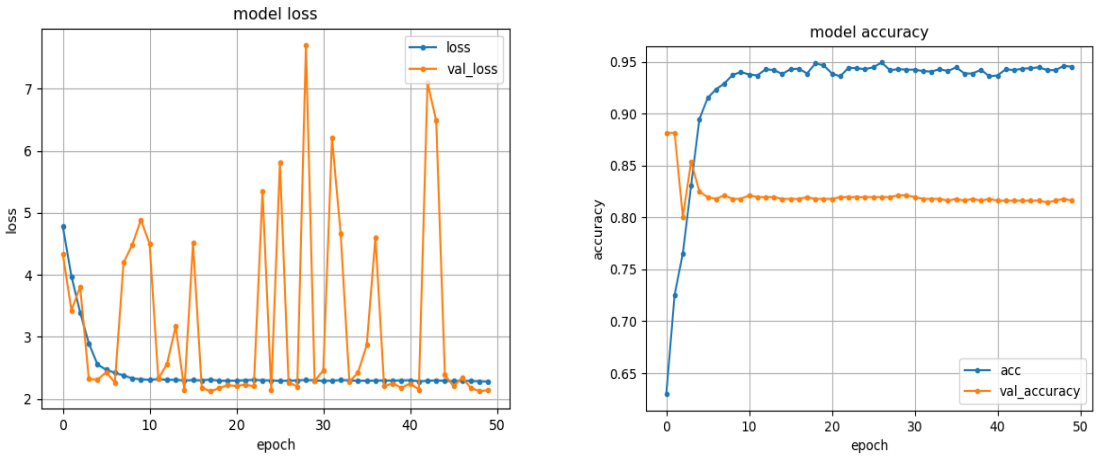
Şekil14- Pooling kaldırılması sonucunda elde edilen sonuçlar

2.2.7. Özgün Ağ Tasarımı

CNN ağ yapısında kaç tane Convolution ve MaxPooling kullanılması gerektiği üzerine belirtilen bir ağ yapısı bulunmamaktadır. Bazı şirketler kendi ağ yapılarını kurarak sunmuştur. Bu ağ yapılarının performansları karşılaştırılarak, her proje için en iyi performans veren ağ yapıları kullanılmaktadır. Projede belirli ağlar üzerinde Convolution parametrelerini, sayısını ve aktivasyon fonksiyonlarını değiştirerek gerekli denemeler yapılmıştır. Aynı zamanda Kullanılan MaxPooling sayısını ve Fully Connected kısmında nöron sayılarını ve katman sayılarını değiştirerek gerekli denemeler yapılmıştır. Denemeler sonucunda iyi sonuç veren bazı ağlar seçilerek buraya eklenmiştir. Aşağıdaki belirtilen özgün ağlar farklı mimarilere sahip CNN ağlarından oluşmaktadırlar.

2.2.7.1. Özgün Ağ 1

Kullanılan ilk özgün ağ tasarımı için elde edilen sonuç grafikleri aşağıdaki gibidir:

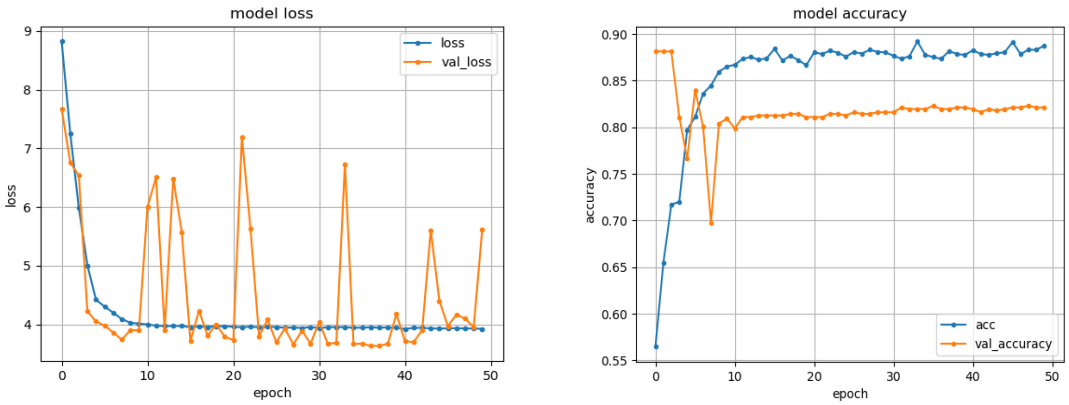


Şekil15- Özgün ağ 1 sonuçları

Yukarıdaki grafiklerde görüldüğü üzere accuracy değeri %95 değerine yakın olmakla birlikte loss değeri 2'ye yakın bir değerdir. Dataset büyütülüp ağ denemeleri yapılırken bu ağ yapısı da kullanılmıştır. Test için loss fonksiyonun sürekli değişmesi, test dataset'in küçük olduğunu belirtmektedir. Projenin geliştirilmesi durumunda dataset genişletilip bu ağ yapısı kullanılabilir.

2.2.7.2. Özgün Ağ 2

Kullanılan ikinci özgün ağ tasarımı için elde edilen sonuç grafikleri aşağıdaki gibidir:

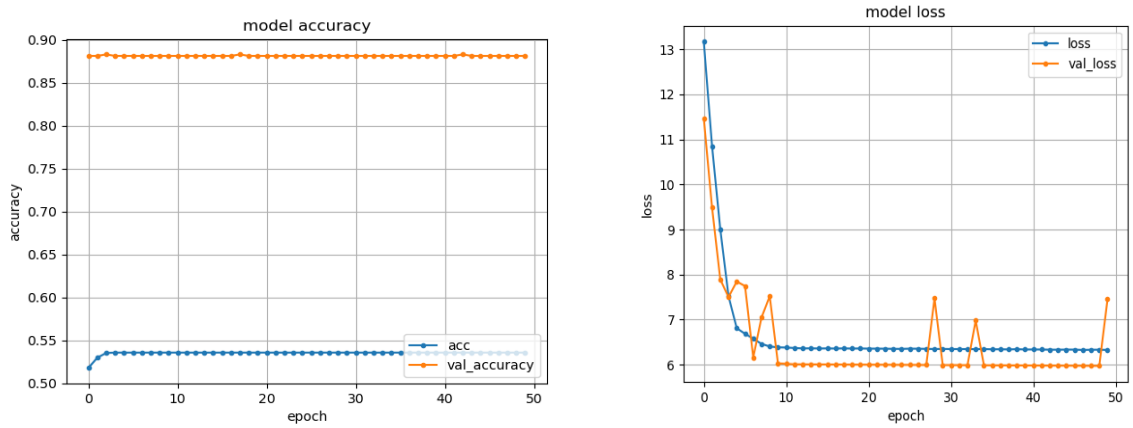


Şekil16- Özgün ağ 2 sonuçları

Yukarıdaki grafiklerde görüldüğü üzere accuracy değeri %90 değerine yakın olmakla birlikte loss değeri 4'e yakın bir değerdir. Projenin geliştirilmesi durumunda dataset genişletilip bu ağ yapısı kullanılabilir.

2.2.7.3. Özgün Ağ 3

Kullanılan ikinci özgün ağ tasarımı için elde edilen sonuç grafikleri aşağıdaki gibidir:



Şekil17- Özgün ağ 3 sonuçları

Yukarıdaki özgün yapısında Convolution sayısı artırılarak deneme yapılmıştır. Bu ağ yapısında eğitim için görüntülenen accuracy %55 değerine yakın ve loss fonksiyonu altıya yakın bir değer olduğundan bu ağ yapısı seçilmemiştir. Projede dataset büyütülmesi durumunda iyi sonuçlar verebileceğinden eklenmiştir.

2.3. Modelin Sonuç Grafikleri

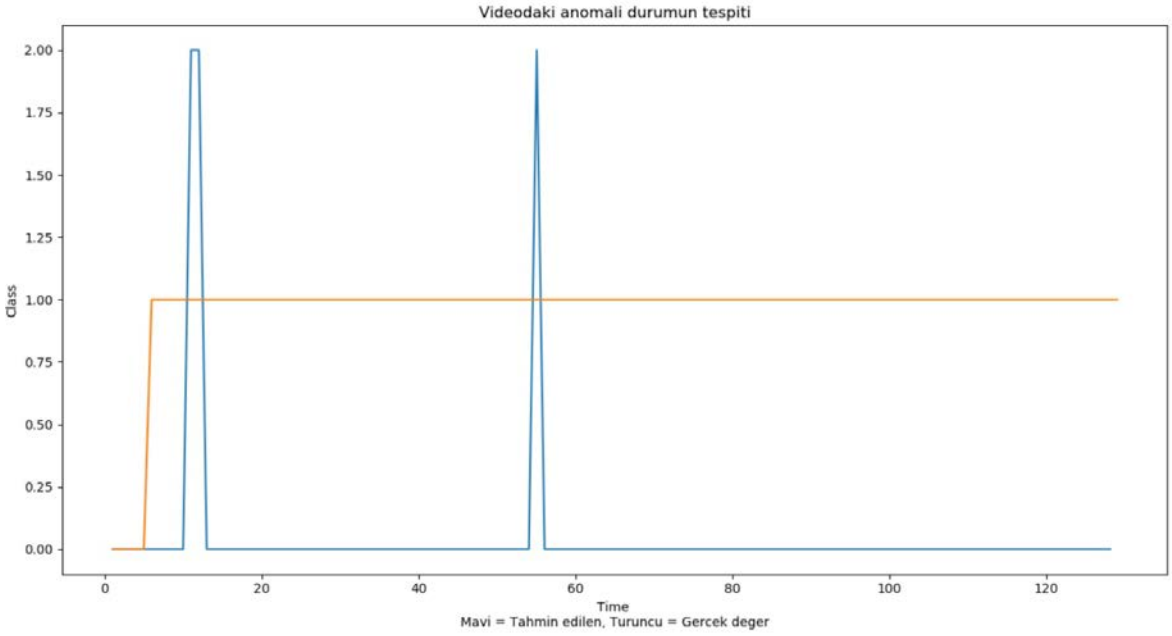
Bu kısımda birinci özgün ağın video ile denenmesi sonucu programın verdiği sonuçlar çizdirilmiştir. Gerekli denemeler ile dataset oluşturulurken ilk olarak video sayısının eşitliği, daha sonra her bir olay için frame sayılarının dağılımlarının eşitliği ve son olarak ağa verilen paketlerin eşitliği denenmiştir. Aynı ağ ve aynı video ile denemeler yapılmıştır. Ağa verilen video dataset içerisindeki bir eğitim ya da test videosu değildir bu nedenle ağın bu videoyu tespit edebilmesi eğitimin iyi gerçekleştiğinin sonucu olmaktadır. Ağ dataset içerisinden verilen videonun tespitini yapabilmektedir çünkü ağa etiketli hallerini verip eğitim yapılmaktadır. Ağa verilen eğitim ya da test videosunun doğru tespit edilmemesi programın yazımında, dataset oluşumunda veya etiketleme kısmında bir hata olduğunu gösterir.

2.3.1. Grafikleri Çizdirme Yöntemi

Grafikler çizdirilirken modelin karar verdiği sonuçlar her 20'lik frame paketi ile alınmış, bir diziye atılmıştır ve bir dosyaya yazılmıştır. Bir dosyadan modelin sonuçlarını ve diğer dosyadan asıl gerçek değerleri için etiketleri okuyan bir Python kodu yazılmıştır. Değerler aynı grafik üzerinde çizdirilmiştir. Aynı grafik üzerinde çizdirilmesinin sebebi sonuçları daha iyi karşılaştırıp irdeleyebilmek içindir.

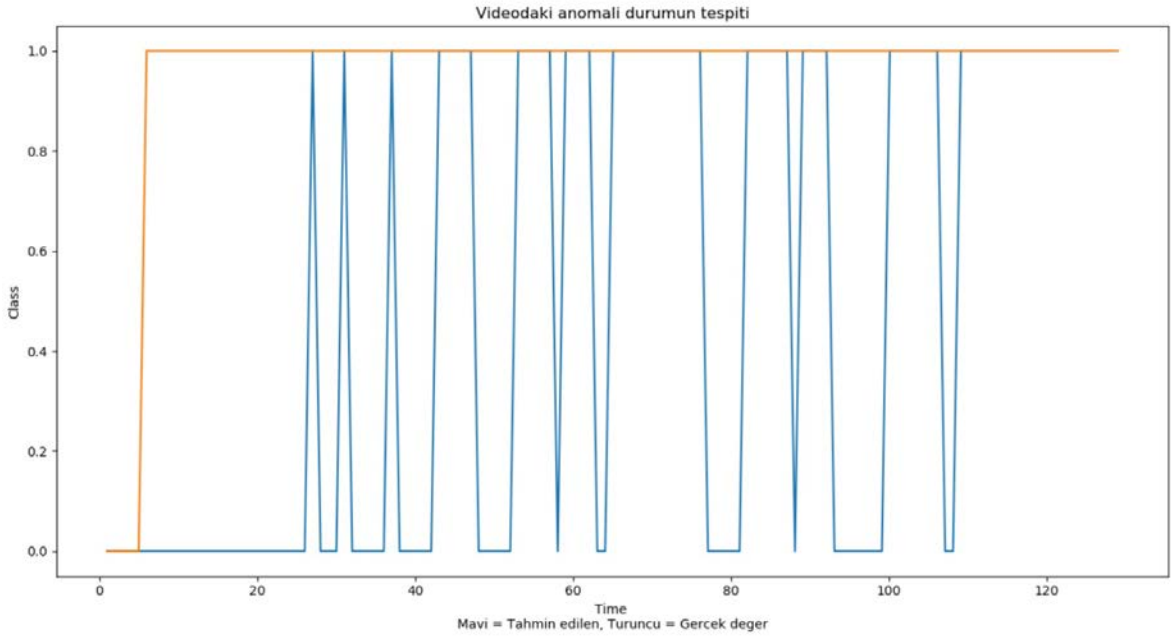
2.3.2. Grafiklerinin Çizdirilmesi ve Yorumlanması

Aşağıdaki grafikte dataset oluşumunda yalnızca video sayısının eşitliği göz önüne alınmıştır. Ağa verilen frame sayısının dağılımı eşit tutulmamıştır veya ağa eşit sayıda frame paketi verilmemiştir. Grafikte görüldüğü üzere programın videonun tahmininde elde ettiği sonuçlar ile gerçek sonuçlar arasında oldukça büyük farklar vardır. Anomali durumun tespiti yapılamamıştır. Gerçek değeri olayın sınıfının 1 olduğunu yani patlama anomali durumunun olduğunu belirtirken modelin ettiği tespitte patlama herhangi bir zamanda gerçekleşmemiştir. Dataset istenildiği kadar büyütülse de, frame sayılarının dağılımları eşitlenmeden ya da ağa eşit sayıda frame paketi verilmeden oluşturulan dataset ile videodaki anomali durumun tespiti iyi sonuç vermeyecektir. Bu nedenle ağa verilen paketlerin sayıları göz önüne alınmalıdır.



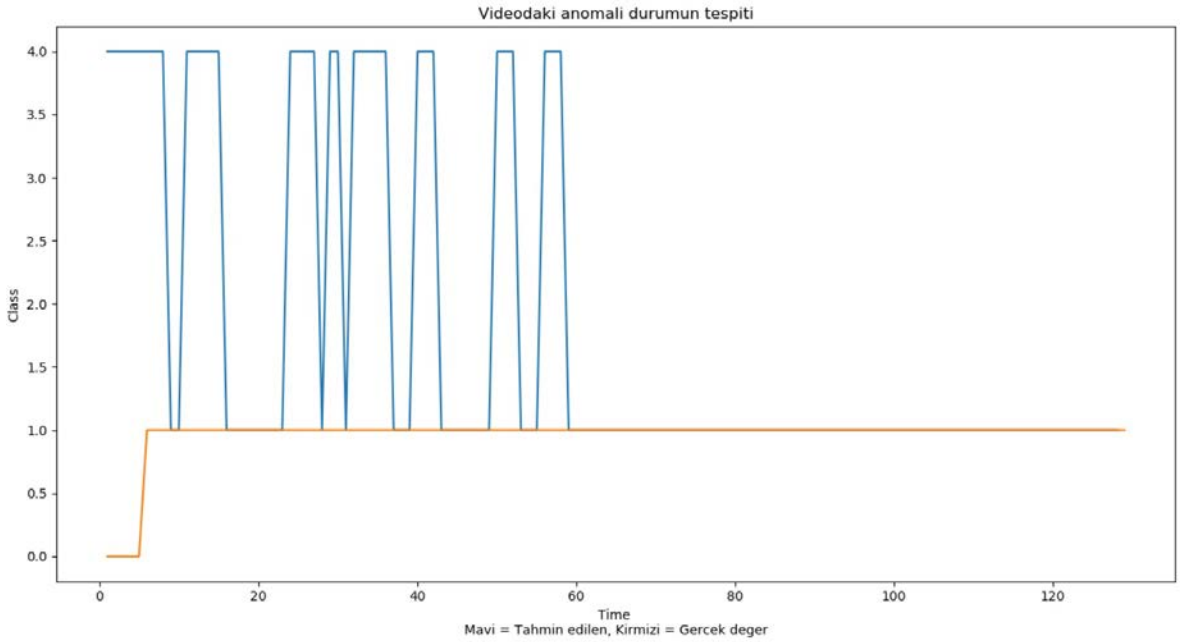
Şekil18- Video sayısının eşit olması durumunda elde edilen tahmin grafiği

Aşağıdaki grafikte dataset oluşumunda her bir olay için ağa verilen framelerin dağılımları eşitlenmiştir ancak burada ağa verilen frame paketlerinin sayıları eşit değildir. Ağa verilen toplam normal olay sayısının çok yüksek olması nedeniyle bazı videoların anomali durumlarının tespiti doğru bir şekilde yapılırken bazı videolar için yanlış yapılmaktadır. Grafikte görüldüğü üzere videodaki olayın gerçek değeri belirli aralıklarda 1 sınıfı yani patlama iken ağın tespit ettiği durum hem patlama hem de normal durum olarak gözlemlenmektedir. Verilecek olan diğer videoların tespitinde de normal olay durumunun her zaman en fazla olacağı göz önüne alınmalıdır. Bunun sebebi ağa verilen frame paketleri için normal olaylı frame paketlerinin çok fazla olmasıdır. Bu durumda ağa verilen frame paketlerinin eşitlenmesi bu problemi çözecektir.



Şekil19- Ağa verilen frame paketlerinin dağılımlarının eşitlenmesi sonucunda tahmin grafiği

Aşağıdaki grafikte dataset oluşumunda ağa verilen frame paketlerinin sayıları eşitlenmiştir ancak dataset boyutu bu işlemle birlikte düşmüştür. Dataset'in büyütülmesi durumunda çok daha iyi sonuçlar gözlemlenilecektir. Grafikte görüldüğü üzere gerçek değeri belirli aralıklarda patlama iken ağın tespiti de benzer şekilde çoğunlukla patlama şeklinde sonuç vermektedir. Bazı insanların patlama esnasında varoluşu yağmalamaya da benzediğinden program aynı zamanda olayı yağmalama olarak da yorumlamıştır. Elde edilen en iyi sonuç aynı ağ için eşit sayıda frame paketi göndermek ile elde edilmiştir.



Şekil20- Ağa verilen frame paketlerinin her sınıf için eşitlenmesi sonucunda tahmin grafiği

2.3.3. Modelin Doğruluk Sonuçları

Grafiklerin çizdirildiği Python kodunda ayrıca etiketlenen 20'lik frame paketleri ile modelin karar vermiş olduğu 20'lik frame paketlerinin sonuçları karşılaştırılmıştır. Yapılan karşılaştırma sonrasında doğruluk sonucu yüzde ile belirtilmiştir.

Ağ Modeli	Her Olay için Ağa Verilen Frame Paketi Sayısı	Her Olay için Ağa Verilen Frame Dağılımı	Doğruluk Oranı
Özgün ağ 1	Eşit değil	Eşit değil	%3.90625
Özgün ağ 1	Eşit değil	Eşit	%55.46875
Özgün ağ 1	Eşit	Eşit değil	%74.21875

Tablo4- Frame paketlerinin farklı dağılımlar sonucunda elde edilen tahmin oranları

2.4 Python ile Basit Bir Yapay Sinir Ağı Kurulması

İlk olarak Fashion MNIST dataset'inden yararlanılarak Python'da basit bir yapar sinir ağı modeli kuruldu. Yapay sinir ağının girişi 28*28 boyutunda bir input almaktadır. Ara katman olarak 128 adet nöron bulunmakta ve çıkış katmanı olarak 10 adet çıkış nöronu

bulunmaktadır. Bu model üzerinde bir eğitim yapılarak ağırlık katsayıları bir *hdf5* uzantılı bir dosyaya kaydedildi. Bu kaydedilen ağırlıkları okunarak bir txt dosyasına kaydedildi.

Daha sonra bu kaydedilen ağırlıklar Python'da yazılan bir kod ile bir txt dosyasına kaydedilmiştir. Bu kaydedilen ağırlıklar bias ve input ağırlıkları olarak ayrılıp başka bir txt dosyasına kaydedildi. İlk katmanda 128 adet nöron bulunduğu için kaydedilen ağırlıkların ilk 128 adeti ilk katmanın bias değerleri, sonraki 28*28*128 = 100352 adet ağırlık ise ilk katmanın ağırlık değerleri olarak başka bir txt dosyasına kaydedildi. Burada 28*28 sayıları input boyutunu temsil etmektedir. Bu inputlardan 128 adet nöronun her birine bir bağlantı olacağından 128 ile de çarpılması gerekmektedir. Bu hesaplamadan sonra 100352 sayısı elde edilmiştir. İlk katmanın ağırlık değerleri ayrıldıktan sonra ikinci katmanın ağırlıklarının ayrılmasına geçildi. İkinci katmanda 10 adet nöron olduğundan kalan ağırlıkların ilk 10 tanesi ikinci katmanın bias katsayıları ve kalan 1280 adet ağırlık ise ikinci katmanın ağırlık katsayıları olarak ayrı bir txt dosyasına kaydedildi. İlk katmanda bulunan 128 adet nöron ikinci katmanın girişi olmaktadır. 128 adet giriş 10 adet nörona gitmektedir. Dolayısıyla 1280 adet bağlantı kurulmaktadır. 1280 sayısı da buradan gelmektedir.

2.4.1. Yapay Sinir Ağının C++ ile Kodlanması

Bu ayırma işlemleri yapıldıktan sonra C++ kısmına geçildi. C++ kısmında ilk olarak input'ların alınması gerekiyordu. MNIST dataset'inden içine aldığı parametre ile istenilen fotoğrafın okunmasını sağlayan bir fonksiyon gerçekleştirildi. Bunun için öncelikle Fashion mnist dataset'inden fotoğrafları almak için Codeblocks projesine import edilmesi gereken dosyalar bulunmaktadır. Bunun için proje dosyalarının içine ".hpp" uzantılı birkaç dosya eklenmesi gerekiyor. Ayrıca fashion mnist dataset'ini yükleyebilmek için 4 adet dosya gerekiyor. Bunların ismi aşağıdaki gibidir:

- t10k-images-idx3-ubyte
- t10k-labels-idx1-ubyte
- train-images-idx3-ubyte
- train-labels-idx1-ubyte

Bu dosyalar projeye dahil edildikten sonra aşağıdaki fonksiyon ile bu dataset'inden istenen fotoğraf okunabilmektedir. İçine aldığı fotoğraf numarasının verilmesi ile istenen fotoğraf sayısal şekilde elde edilmektedir.

Ayrıca bu fotoğrafı input adında iki boyutlu bir pointer diziye atılmıştır. C'de 2 boyutlu pointer kullanmak için ilk olarak bellekte allocate işlemi yapılması gerekmektedir. Bunun için ayrıca iki_boyutlu isminde bir fonksiyon yaratılmıştır. Bu fonksiyon bellekte kendisine verilen uzunluklarda 2 boyutlu bir dizi oluşturmaktadır.

```
float** fashion_mnist_oku(int goruntu)
{
    float** input = iki_boyutlu(28, 28);
    mnist::MNIST_dataset<std::vector, std::vector<uint8_t>, uint8_t> dataset =
                            mnist::read_dataset<std::vector, std::vector, uint8_t,
                    uint8_t>(MNIST_DATA_LOCATION);
    for (int row = 0; row < 28; row++)
    {
        for (int column = 0; column < 28; column++)
        {
            input[row][column] = unsigned(dataset.test_images[goruntu][(row * 28 +
                                                                        column)]);
        }
    }
    return input;
}
```

Bu fonksiyon main kısmında şu şekilde çağırılarak istenilen fotoğrafın matris şeklinde okunması sağlanmıştır.

```
int main(){
float** image = fashion_mnist_oku(0);
…
}
```

İstenen fotoğrafın okunmasından sonra alınan bu fotoğrafın yapay sinir ağından çıkan ağırlıklar ile işlemler yapılarak Python tarafında elde edilen sonuçlarla aynı sonucun elde edilmesi gerekmektedir. Fakat öncelikle farklı txt dosyalarına ayrılan ağırlıkların C++ kısmında okunması bir diziye atanması gerekiyor. Bunun için *Weights* adında bir class oluşturarak bu gerçekleştirildi.

Weigths class'ının yapısı aşağıdaki gibidir

```
6
7   class Weights
8   {
9       public:
10          Weights(int firstLayerBiasSize, int firstLayerKernelSize, int secondLayerBiasSize, int secondLayerKernelSize);
11          void setFirstLayerWeight(string nameOfTxt);
12          float getFirstLayerWeight(int index);
13
14          void setSecondLayerWeight(string nameOfTxt);
15          float getSecondLayerWeight(int index);
16
17          void setFirstLayerBias(string nameOfTxt);
18          float getFirstLayerBias(int index);
19
20          void setSecondLayerBias(string nameOfTxt);
21          float getSecondLayerBias(int index);
22
23          virtual ~Weights();
24      private:
25          float *first_layer_bias;
26          float *first_layer_weights;
27          float *second_layer_bias;
28          float *second_layer_weights;
29
30  };
```

Şekil21- Weights class yapısı

Bu ağırlıklar okunduktan sonra oluşturulacak nöronların tutulacağı bir class daha oluşturma gereği duyulmuştur. Bu maksatla *Neuron* adında bir class yapısı oluşturuldu.

Neuron class'ının yapısı aşağıdaki gibidir:

```
5   class Neuron
6   {
7       public:
8           Neuron();
9           void setId(int id);
10          int getId();
11
12          void setValue(float w, int id);
13          float getValue(int i);
14          virtual ~Neuron();
15
16      private:
17          int id;
18          float  value;
19          Neuron* next;
20          Neuron* head;
21
22  };
```

Şekil22- Neuron class yapısı

Neuron Classı'nda her bir nöron bağlı bir liste yapısıyla tutulmuştur. Hesaplanan nöronlar sırasıyla kuyruğa eklenerek bir sıralı liste oluşturulmuştur.

Bütün bu gerekli class yapıları oluşturulduktan sonra aşağıdaki şekilde main'den çağırılarak kullanılmıştır. Böyle yapılmasının amacı kodun daha okunabilir ve temiz bir şekilde yazılmasını sağlamaktır. Kod yazılırken gerekli yazılım mühendisliği kurallarına uyulmasın özen gösterilerek yazılmıştır

```
Weights weights(128, 100352, 10, 1280);

weights.setFirstLayerBias("bias_1.txt");
weights.setSecondLayerBias("bias_2.txt");
weights.setFirstLayerWeight("first_weights.txt");
weights.setSecondLayerWeight("second_weights.txt");
```

Bu setleme işlemlerinden sonra alınan fotoğraf ile txt dosyalarından okunan ağırlıklar çarpılıp toplanarak yapay sinir ağının çıktıları alınmıştır. Bu çıktılar Python ile elde edilen çıktılar ile karşılaştırılarak çıktının doğru olup olmadığı kontrol edilmiştir. Kontroller sonucunda aynı çıktıların alındığı gözlemlenmiştir.

2.4.2. Conv2D ile Yapay Sinir Ağının Kodlanması

Python'da *Cifar10* dataset'ini kullanarak bir yapay sinir ağ modeli kuruldu. Bu yapay sinir ağ modeli kurulduktan sonra eğitiminin yapılması sağlandı. Eğitim yapıldıktan sonra ağırlıkları *hdf5* uzantılı olarak kayıt edildi. Cifar10 dataset'inde aşağıdaki sınıflar yer almaktadır. Python kodunda bu belirtilmiştir.

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
        'dog', 'frog', 'horse', 'ship', 'truck']
```

Modelin ağ yapısı aşağıdaki gibidir:

```
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Yapay sinir ağının modeli de aşağıdaki gibidir:

```
model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(10, activation='softmax'))
```

Bu kısımdan sonra eğitim yapılarak *hdf5* olarak kayıt edildi. Bu kayıt işleminden sonra ağırlıkların bulunduğu *hdf5* dosyası içeriği okunarak ağırlıkları txt dosyasına alınmıştır. Bu ağırlıklar elde edildikten sonra tek tek katmanların isminde txt dosyalarına kayıt edilerek C++ ile kodlandığı esnada okunmak amacıyla ayrılmıştır.

Python kısmında işlemler tamamlandıktan sonra C++ kısmında bu ağın kodlanmasına geçildi. Fashion mnist dataset'indeki okuma işlemi gibi benzer şekilde proje dosyasına ilk olarak Cifar10 dataset'inin okunması için gerekli dosyalar eklenmiştir. Daha sonra Cifar10 dataset'inde bulunan fotoğrafların okunması gerçekleştirildi. Fonksiyona verilen integer görüntü numarasına göre dataset'inden o numaraya ait fotoğraf elde edilmektedir. Fonksiyon aşağıdaki gibidir:

```
float*** cifar_10_oku(int goruntu)
{
    float*** input = uc_boyutlu(32, 32, 3);
    auto dataset = cifar::read_dataset<std::vector, std::vector, uint8_t,
uint8_t>();
    int r, g, b;
    for (int row = 0; row < 32; row++)
    {
        for (int column = 0; column < 32; column++)
        {
            r = unsigned(dataset.training_images[goruntu][(row * 32 + column)]);
            g = unsigned(dataset.training_images[goruntu][(row * 32 + column) +
1024]);
            b = unsigned(dataset.training_images[goruntu][(row * 32 + column) +
2048]);
            input[row][column][0] = r;
            input[row][column][1] = g;
            input[row][column][2] = b;
        }
    }
    return input;
}
```

Fonksiyonda görüntünün okunması için 3 boyutlu bir pointer dizinin kullanılması gerekmektedir. Bunun için bellekte 3 boyutlu bir dizinin allocate edilmesi amaçlı

uc_boyutlu() fonksiyonu yazılmıştır. Bu fonksiyon kullanılarak bellekte görüntünün yerleştirileceği yer ayrılmıştır ve görüntü bu diziye aktarılmıştır. Fonksiyon yazıldıktan sonra main kısmında fotoğrafın okunması sağlanmıştır.

```
int main(){
float*** resim = cifar_10_oku(7);
…
}
```

Bu okuma işlemi gerçekleştirildikten sonra *Conv2D* kodlanmasının daha anlaşılır ve kolay yapılması açısından bir class yapısı oluşturulmuştur. Bu class'ın header dosyası aşağıdaki gibidir:

```
4   #include <stdlib.h>
5   #include <stdio.h>
6   using namespace std;
7   class Conv2D
8   {
9       public:
10          Conv2D();
11          float getLayerWeights(int i, int j, int m, int n);
12          void setLayerArrayAllocate(int a, int b, int c, int d);
13          void setLayerWeights(string fileName, int x, int y, int z, int t);
14          void setBias(string fileName, int size);
15          float getBias(int i);
16          virtual ~Conv2D();
17
18      private:
19          float *bias;
20
21          float ****layerWeights;
```

Şekil23- Conv2D Class'ının header dosyası

Burada yer alan fonksiyonların ne amaçla kullanıldıkları aşağıda verilmiştir.

`setLayerArrayAllocate()`: Bu fonksiyon 4 boyutlu bir matris allocate etmek için kullanılmaktadır. Class'ta bulunan *layerWeights* isimli pointer'ın allocate edilmesini sağlamaktadır. İçeriğine aldığı parametreler ise boyutlarını içten dışa doğru boyutlarını ifade etmektedir.

`setLayerWeights()`: Bu fonksiyon Conv2D katmanının ağırlıklarını txt dosyasından okunmasını ve layerWeights dizisine atılmasını sağlamaktadır.

`getLayerArrayWeights()`: Bu fonksiyon içine aldığı parametrelerde bulunan adresteki ağırlık değerini elde etmek için kullanılmaktadır.

`setBias():` Bu fonksiyon o katmanın bias değerlerini txt dosyasından okunmasını ve bias dizisine atılmasını sağlamaktadır.

`getBias():` Bu fonksiyon içine aldığı adresteki bias değerini elde etmek için kullanılmaktadır.

Conv2D Class'ını implement ettikten sonra main kısmında kullanıp ağırlıkların okunması sağlanmıştır.

```
Conv2D conv;
conv.setLayerArrayAllocate(3, 3, 3, 32);
conv.setLayerWeights("conv_kernel1.txt", 3, 3, 3, 32);
conv.setBias("conv_bias1.txt", 32);
float*** resim = cifar_10_oku(7);
```

Ağırlıkların okunması gerçekleştirildikten sonra Convolution katmanının kodlanması gerçekleştirilmiştir. İlk olarak ağın test edilmesi için Cifar10 dataset'inden bir fotoğraf okunarak ağa verilmiştir. Dataset'ten 7 numaralı fotoğraf üstteki şekilde okunmuştur. Bu fotoğraf 32*32*3 boyutlu bir fotoğraftır. Bu fotoğrafa *Convolution* uyguladıktan sonra çıkan matrisin boyutu 30*30*32 olmaktadır. Convolution katmanında filtre boyutu 3*3 olduğundan boyut 30*30'a düşmüştür. Diğer yanında bulunan 32 sayısı ise nöron derinliğini ifade etmektedir. Bu katmandan sonra *MaxPooling* katmanı vardır. Convolution katmanından alınan değerler bir dizide tutulmaktadır. Bu dizide bulunan değerlere MaxPooling uygulanmıştır. MaxPooling uygulandıktan sonra çıkış dizisinin boyutu yarı yarıya düşmüştür. Yani sonuç olarak çıkış dizisi 15*15*32 olmuştur.

MaxPooling katmanından sonra ağ yapısının devamında *Conv>MaxPooling>Conv* katmanları yer almaktadır. Üstte anlatılan şekilde bu katmanların da kodlanması gerçekleştirilmiştir. CNN katmanlarından çıktıktan sonra 4*4*64 boyutunda bir dizi elde edildi. Sıradaki işlem ise bu çıkış dizisinin Full Connected yapılarak yapay sinir ağına verilmesidir. Bunun için çıkış dizisi tek boyutlu bir dizi haline getirmek amaçlı *flatten()* fonksiyonu yazılmıştır.

```
float* flatten(float*** input, int width, int height, int depth)
{
    int arraySize = width * height * depth;
    float* flattenArray = new float[arraySize];
    int index = 0;
    for(int i=0; i<width; i++)
    {
        for(int j=0; j<height; j++)
        {
            for(int k=0; k<depth; k++)
            {
                flattenArray[index++] = input[i][j][k];
            }
        }
    }
    return flattenArray;
}
```

Bu fonksiyon kendisine verilen 3 boyutlu bir pointer dizisinin *width, height* ve *depth* boyutlarını alarak *width * height * depth* boyutunda tek boyutlu bir dizi oluşturmaktadır. Oluşturduğu bu diziye input dizisinden aldığı değerleri atmaktadır.

Flatten kısmından sonra ise elde edilen çıkışın yapay sinir ağlarına verilmesine sıra geldi. Bunun için daha oluşturulan *Weights* class'ı kullanılmıştır. Bu class yardımıyla katmanların ağırlıkları okunmuştur.

```
Weights convWeights(64, 65536, 10, 640);

convWeights.setFirstLayerBias("fully_bias_1.txt");

convWeights.setFirstLayerWeight("fully_kernel_1.txt");

convWeights.setSecondLayerBias("fully_bias_2.txt");

convWeights.setSecondLayerWeight("fully_kernel_2.txt");
```

Bu atamalar gerçekleştirildikten sonra elde edilen ağırlıklar ile *flatten* fonksiyonundan elde edilen tek boyutlu matris Yapay Sinir Ağının giriş katmanının nöronlarını oluşturmaktadır. Daha sonra bu giriş matrisi ile okunan ağırlık değerleri gereken şekilde çarpılıp toplanmasının ardından ilk katmanda 64 adet bulunan nöronlara aktarılmıştır. Bu kısımdan sonra katmanın ikinci kısmı kodlanmıştır. Çıkış katmanında ise 10 adet nöron bulunmaktadır. Çıkışta 10 adet nöronun olması ağda 10 farklı sınıfın olduğunu göstermektedir. İlk katmanda bulunan nöronlar ikinci katmanın giriş değerleri olmaktadır. İlk katmandan elde edilen giriş değerleri ile ikinci katmanın ağırlıkları da çarpılıp toplanarak çıkış katmanının değerleri elde edilmiştir. Bu kodlama gerçekleştirildikten sonra Python'da

test sonucu elde edilen sonuçlar ile karşılaştırılmıştır. Aynı sonuçların alındığı gözlemlenmiştir. Ayrıca dikkat edilmesi gereken konulardan biri, Python kısmında test işlemini yaparken aktivasyon fonksiyonları kaldırılarak test işlemi yapılmaktadır. Burada kullanılan ağda aktivasyon fonksiyonu olarak *relu* kullanılmıştır. Test kısmında ise bu fonksiyon kaldırılıp, test işlemi yapılmıştır.

2.4.3. Conv3D ile Yapay Sinir Ağının Kodlanması

Bu bölümde ise Conv3D ile kurulan bir ağdan elde edilen ağırlıkların, giriş olarak verilen videoya bazı işlemler ile uygulanması sonucu elde edilen çıkış değerinin, Python ile elde edilen çıkış değeri ile karşılaştırılması gerçekleştirilmiştir. İlk olarak Python'da basit bir Conv3D ağı kurulmuştur ve bu ağ ile eğitim yapılmıştır. Ağ yapısı aşağıdaki gibidir:

```
model = keras.models.Sequential()

model.add(Conv3D(8, kernel_size=(5, 5, 3), activation="relu",
            kernel_regularizer=keras.regularizers.l2(normalizasyon_katsayisi),
                                                    input_shape=input_shape))
model.add(MaxPool3D((2, 2, 1), strides=(2, 2, 1)))

model.add(Conv3D(16, kernel_size=(5, 5, 3), activation="relu",
             kernel_regularizer=keras.regularizers.l2(normalizasyon_katsayisi)))

model.add(MaxPool3D((2, 2, 2), strides=(2, 2, 2)))
```

Bu basit olan ağ kurulduktan sonra ardından Yapay Sinir Ağı oluşturulmuştur. Bu ağ yapısı aşağıdaki gibidir:

```
model.add(Dense(32, activation='relu',
            kernel_regularizer=keras.regularizers.l2(normalizasyon_katsayisi)))

model.add(Dropout(0.6))

model.add(Dense(16, activation='relu',
            kernel_regularizer=keras.regularizers.l2(normalizasyon_katsayisi)))
```

```
model.add(Dropout(0.6))
model.add(Dense(1, activation='sigmoid'))
```

Görüldüğü üzere ilk katmanda 32 adet ikinci katmanda 16 adet nöron bulunmaktadır. Çıkış ise tek nöron ile ifade edilmiştir. Bu ağ ile bir eğitim yapıldıktan sonra diğerlerindeki gibi bir *hdf5* dosyasına kayıt edilmiştir. Daha sonra bu dosya içeriği okunarak txt dosyasına çevrilmiştir. Bu txt dosyasında bulunan değerler her katmanın kendi ağırlıkları olarak ayrı ayrı txt dosyalarına kayıt edilmiştir. Diğer kısımlarda yapıldığı gibi Conv3D için de aynı şekilde bir class oluşturulmuştur. Bu class yapısı aşağıdaki gibidir:

```
5   #include <stdio.h>
6   using namespace std;
7
8   class Conv3D
9   {
10      public:
11          Conv3D();
12          float getLayerWeights(int i, int j, int m, int n, int t);
13          void setLayerArrayAllocate(int a, int b, int c, int d, int e);
14          void setLayerWeights(string fileName, int x, int y, int z, int t, int s);
15          void setBias(string fileName, int size);
16          float getBias(int i);
17          virtual ~Conv3D();
18
19      protected:
20
21      private:
22           float *bias;
23
24          float *****layerWeights;
25  };
```

Şekil24- Conv3D Class'ının header dosyası

Burada yer alan fonksiyonlar Conv2D'de bulunan fonksiyonlar ile aynı işi yapmaktadır. Burada tek fark olarak ağırlıkların 5 boyutlu bir pointer dizisine atılmasıdır.

Sıradaki işlem ise ağırlıkların C++ tarafında okunarak girişine verilen videoya uygulanmasıdır. Videonun ağırlık piksel değerleri bir txt dosyasına aktarılmıştır ve buradan okunarak işlemler gerçekleştirilecektir. Ağırlıkların ve videonun txt dosyalarından okunması aşağıdaki gibi gerçekleştirilmiştir:

```
Conv3D conv3d;
    conv3d.setLayerArrayAllocate(5, 5, 3, 3, 8);
    conv3d.setLayerWeights("conv3d_kernel_1.txt", 5, 5, 3, 3, 8);
```

```
    conv3d.setBias("conv3d_bias_1.txt", 8);

fstream oku("c3d_input.txt");
    string row;
    float input[32][32][10][3];
      …
```

fstream ile input dosyası okunup 4 boyutlu input dizisine atılmıştır. Bu işlemler gerçekleştirildikten sonra sırada ağ yapısının kodlanması gelmektedir. Giriş videosu üstteki kodda görüldüğü gibi 32*32*10*3 boyutundadır. Bu input'a Convolution uygulandıktan sonraki boyutu ise 28*28*8*8 olmaktadır. Çünkü Convolution'da bulunan kernel matrisinin boyutunun 5*5*5 olmasından kaynaklıdır. 28*28*8*8 boyutlu dizinin sonunda bulunan 8 değeri katmandaki nöron sayısını ifade etmektedir. Convolution'dan elde edilen çıkış matrisi, ağdaki sıra takip edildiğinde sırada MaxPooling kısmına giriş olarak verilmiştir. MaxPooling girişine uygulanan matrisi çıkışta boyutunu yarı yarıya düşürmektedir. Bu ise içine aldığı kernel matrisinin boyutuyla alakalıdır. Burada 2*2*2 boyutlu bir kernel matrisi olduğu için dizinin boyutu yarısına düşmektedir. Bu yüzden bu katmanın çıkışı 14*14*8*8 olmaktadır. Ek olarak kernel matrisi boyutunun 2*2*1 olması durumunda fotoğrafın yükseklik ve genişlik değerleri yarıya düşmekte ancak derinliği ifade eden frame paketi grubu yani gönderilen fotoğraf sayısı yarıya düşmemektedir. Bu kısımdan sonra geriye *Conv>MaxPooling* katmanları kalmaktadır. Bu katmanlar da üstte belirtildiği gibi kodlanması gerçekleştirilmiştir. Sonuç olarak en son MaxPooling'ten çıkış dizisi 5*5*3*8 olmaktadır.

CNN katmanlarından çıkan bu çıkış değeri *flatten3d()* fonksiyonu ile tek boyuta indirgenerek yapay sinir ağına giriş olarak verilecektir. Bu yüzden, kendisine verilen 4 boyutlu bir diziyi tek boyuta çeviren *flatten3d()* adında bir fonksiyon yazılmıştır. Fonksiyon aşağıdaki gibidir:

```
float* flatten3d(float**** input, int width, int height, int depth, int frame)
{
    int arraySize = width * height * depth * frame;
    float* flattenArray = new float[arraySize];
    int index = 0;
    for(int i=0; i<width; i++)
    {
        for(int j=0; j<height; j++)
        {
            for(int k=0; k<depth; k++)
            {
                for(int t=0; t<frame; t++)
                {
                    flattenArray[index++] = input[i][j][k][t];
                }
            }
        }
    }
    return flattenArray;
}
```

Fonksiyona verilen dizinin *width, height, dept* ve *frame* sayılarının çarpımı boyutunda bir tek boyutlu dizi elde edilmiştir. Yine parametre olarak aldığı *input* dizisinin içeriğini bu oluşturulan tek boyutlu diziye aktarılma işlemi yapılmıştır. Bu şekilde Full Connected olarak ağa verilme işlemine hazırlanmış olmaktadır.

Ağdaki ilk katmanda 32 nöron, ikinci katmanında 16 nöron ve çıkışta da 1 nöron bulunmaktadır. Ağ ağırlıklarının setlenmesi için *Weights* class'ından yararlanılmıştır. Ayrıca nöronların değerlerinin tutulması için de *Neuron* class'ından faydalanılmıştır.

```
Weights convWeights(64, 65536, 10, 640);
convWeights.setFirstLayerBias("fully_bias_1.txt");
convWeights.setFirstLayerWeight("fully_kernel_1.txt");
convWeights.setSecondLayerBias("fully_bias_2.txt");
convWeights.setSecondLayerWeight("fully_kernel_2.txt");

float firstTemp[64] = { 0 } , secondTemp[10] = { 0 };

Neuron firstLayerNeuron;
Neuron secondLayerNeuron;
```

Burada bütün katmanların ağırlıklarının atanması gerçekleştirilmiştir. Birinci katmanın nöron değerleri *firstLayerNeuron* nesnesinde saklanmaktadır. İkinci katmanın

nöron değerleri ise *secondLayerNeuron* nesnesinde saklanmaktadır. İlk katmanda okunan ağırlıklar *flatten3d* fonksiyonundan gelen tek boyutlu diziyle çarpılıp toplanarak ilk katman nöronlarının değerleri elde edilmiştir. İkinci katmanda ise bu ilk katmandan elde edilen nöron değerleri ile ikinci katmandaki ağırlık değerleri çarpılıp toplanarak ikinci katmandaki nöronlarının değeri elde edilmiştir. Son olarak çıkış nöronunu hesaplamak için son katmandaki ağırlıklar ile ikinci katmanın nöronları çarpılıp toplanarak çıkış nöronunun değeri elde edilmiştir. Python kısmında yapılan testte elde edilen değerler ile ara nöron ve çıkış nöronlarının çıktıları karşılaştırılmıştır ve doğru sonuçlar elde edilmiştir.

2.4.4. Conv3D ile Çıkıştaki Sınıf Sayısının Artırılması

Conv3D kısmında yapılan örnekte tek çıkışlı bir ağ yapısı kullanılmıştır. Bu ağ yapısında çıkışta aktivasyon fonksiyonu olarak *sigmoid* fonksiyonu kullanılmıştır. Buradaki ağın çok sınıflı hale getirmek için çıkıştaki aktivasyon fonksiyonunun *softmax* ile değiştirilmesi gerekmektedir. Sigmoid fonksiyonu genellikle iki adet çıkışı bulunan ağlarda kullanılmaktadır. Bu fonksiyon kullanıldığında çıkışta 2 adet değer üretilmektedir. Bu değer -1 ile 1 ile temsil edilmektedir. Softmax ise çıkışta istenilen sayıda çıkış değeri üretmeyi sağlar. Bu fonksiyon -1 ile 1 arasında bütün reel sayıları üretmektedir. Anlaşıldığı üzere ikiden çok sınıf sayısına sahip ağ yapılarının çıkış katmanında softmax kullanılması daha iyi sonuç vermektedir. Python'daki kod kısmında son bölümde Dense ile ifade edilen ağın son katmanında sınıf sayısı belirtilmektedir. Sınıf sayısının artırılması durumunda içerisine yazılan nöron sayısı değiştirilmelidir. C++ kısmında ise ufak bir değişiklikle çok sınıflı bir yapının çıkış değerleri hesaplanabilir. C++ kodlanmasında çıkışı tek bir değişkenle ifade etmek yerine o değişkeni dizi ile değiştirerek üretilen çıkışlar hesaplanarak dizide tutulabilir.

2.5. C++ Kodların Paralelleştirilmesi

2.5.1. OpenMP nedir?

OpenMP Solaris, IBM AIX , HP-UX, GNU/LINUX, MAC OS X ve Windows işletim sistemleri üzerinde çoğu işlemci mimarisi üzerinde Fortran, C++, C programlama dillerinde çoklu platform paylaşımlı bellek çoklu işlemeyi destekleyen bir uygulama geliştirme ara yüzüdür, yani bir API'dir. OpenMP derleyici yönergelerinin kütüphane rutinlerini ve ortam değişkenlerinin çalışma zamanı davranışını etkileyen bir kümesini içerir.

OpenMP çoklu iş parçacığı gerçekleştirimidir. Çoklu iş parçacığı ana iş parçacığının(sırasıyla yürütülen komutların bir dizisi) belirli bir sayıda yardımcı iş parçacıklarını durdurması ve bir görev onlar arasında paylaştırması olan paralelleştirme metodudur. İş parçacıkları birbiri ardında paralel şekilde çalışırlar. Farklı işlemcilerin iş parçacıkları farklı çalışma zamanı ortamlarını kendilerine tahsis ederler.

Paralel çalışacak olan kodun bir bölümü sırasıyla işaretlenir. Bu işaretleme kod bölümünün yürütülmesinden önce iş parçacıklarının o kod bölümüne girmelerine sebep olacak ön işlemci direktifleridir. Her bir iş parçacığı onlara bağlı bir ID'ye sahiptir. *omp_get_thread_num()* fonksiyonu ile bu ID elde edilir. İş parçacığı ID'si bir tam sayıdır ve ana iş parçacığının ID'si sıfırdır. Paralelleştirilmiş kodun yürütülmesinden sonra iş parçacıkları ana iş parçacığına tekrar geri katılırlar. Programın sonuna kadar bu böyle devam eder.

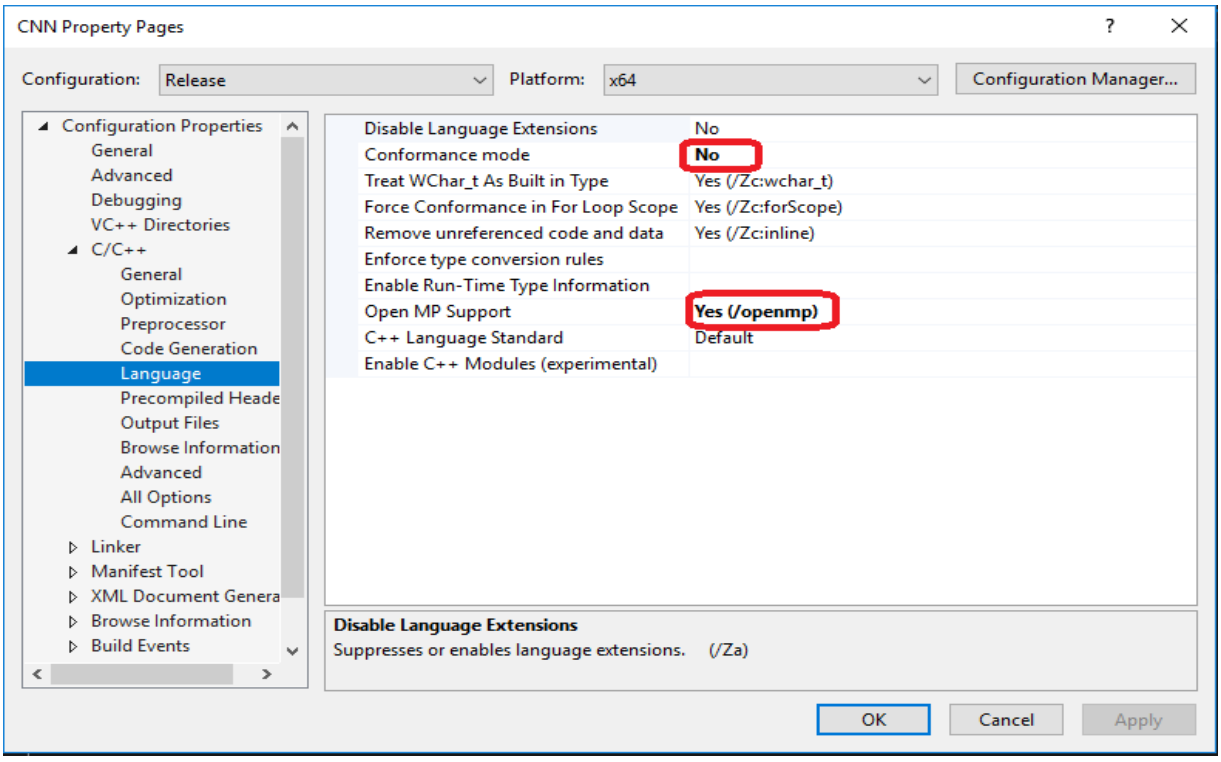
Varsayılan olarak her bir iş parçacığı kodun paralelleştirilmiş bölümünü birbirinden bağımsız şekilde yürütür. İş paylaşımı yapıları iş parçacıkları arasında bir görevi paylaştırmak için kullanılabilir, böylece her bir iş parçacığı kodun kendisine ayrılan bölümünde çalışır. Hem görev paralelleştirme hem de veri paralelleştirme OpenMP kullanarak bu yöntemle yapılır.

Çalışma zamanı ortamı kullanıma bağlı olarak işlemcilere iş parçacığı tahsis eder, makine yükleme ve diğer faktörler gibi. İş parçacıklarının sayısı ortam değişkenleri veya kod içerisinde kullanılan fonksiyonlara bağlı olarak çalışma zamanı ortamı tarafından atanır. OpenMP fonksiyonları *omp.h* etiketli C/C++ header dosyaları ile programa dahil edilir.

2.5.2. Conv3D Kodlanmasının Paralelleştirilmesi

OpenMP'yi kullanmak için ilk olarak CodeBlocks'ta kodlanması gerçekleştirilen kodun Visual Studio ortamına taşınması gerekiyordu. İlk olarak Visual Studio ortamına kodların taşınması gerçekleştirilmiştir. Bu kodlar taşınırken birkaç optimizasyon da yapılmıştır. Birkaç kez tekrarlanan işlemlerin fonksiyonlar ile gerçekleştirilmesi yapılmıştır. Bu şekilde kod kalabalığı biraz da olsa azaltılmıştır.

Visual Studio OpenMP ayarlamaları gerçekleştirildi. Projenin *Properties* kısmına tıklandığında aşağıdaki pencere ile karşılaşılmaktadır. Bu pencerede görülen adımlar takip edilerek ve işaretlenen ayarlar şekildeki gibi yapıldığında OpenMP projede uygulanmaya hazır olmaktadır.



Şekil25- OpenMP ayarlarının yapılması

Bu ayarlama yapıldıktan sonra geriye sadece projeye OpenMP'nin header dosyasının eklenmesi kalmaktadır. OpenMP header dosyası aşağıdaki gibi eklenmektedir.

```
#include <omp.h>
```

OpenMP'nin döngüler üzerinde kullanılan bir fonksiyonu ile bu paralelleştirme gerçekleştirilmiştir. Aşağıda Convolution fonksiyonuna uygulanan OpenMP kodu yer almaktadır.

```
float**** convolution(float**** input, Conv3D conv3d, int noron_size, int frame_size,
                                                    int  width, int  height, int depth ) {

    int bb = 0;
    int cc = 0;
    int dd = 0;
    float**** result_conv = arrayAllocate3d(width - 4, height - 4, frame_size,
                                                                    noron_size);

#pragma omp parallel for schedule(static) shared(input, conv3d)
    for (int a = 0; a < noron_size; a++)  // nöron sayısı
    {
        bb = 0;
        for (int b = 1; b < frame_size - (1); b++)///input frame sayısı
        {
...
```

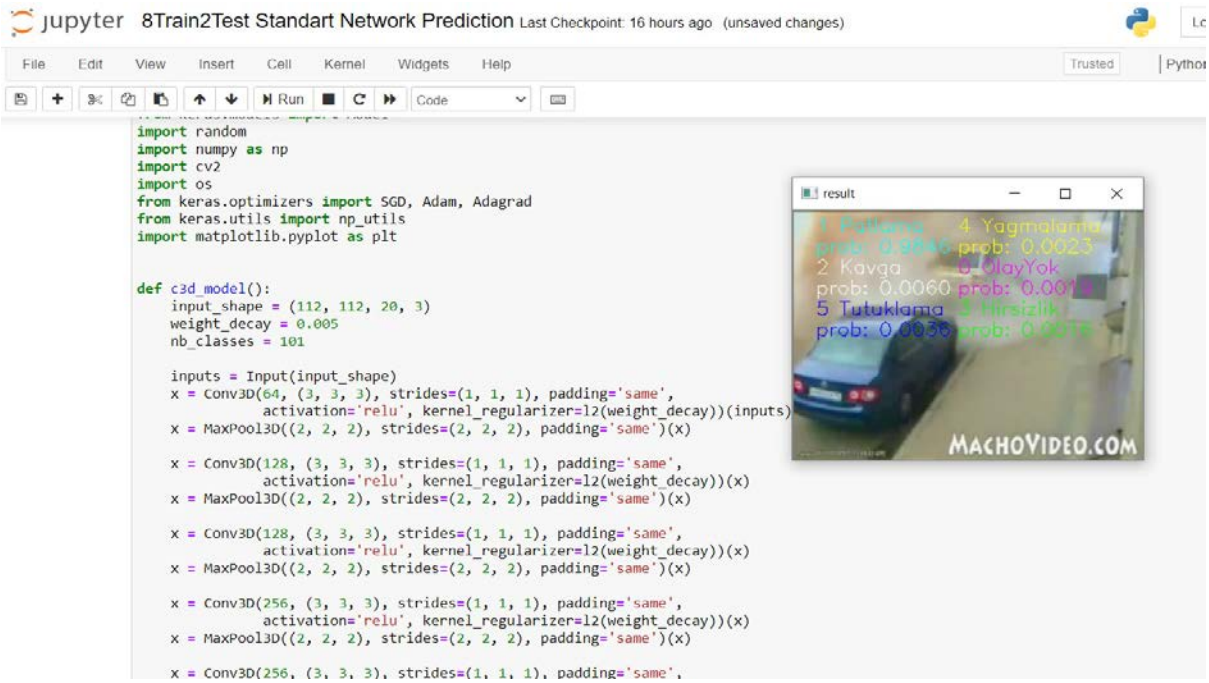
Burada *#pragma omp paralel* ile başlayan kısım kodu paralel şekilde çalışmasını sağlamaktadır. *Schedule* kısmı ise içine aldığı parametreye göre *for* döngüsünü belli bloklara bölerek paralelleştirme yapmaktadır. Burada içine verilen *static* parametresi for döngüsünün iterasyon sayısını *thread* sayısına bölerek her thread'e eşit şekilde paylaştırmaktadır. *Shared* kısmında ise *input* değişkeni ve *conv3d* nesnesinin thread'ler arasında paylaşımda olduğunu thread'lere bildirmek amaçlı kullanılmaktadır. Bu fonksiyon *MaxPooling* fonksiyonuna da uygulanmıştır. Bu şekilde programın paralelleştirilmesi sağlanmıştır.

3. Sonuçlar

Elde bulunan dataset'ten 5 adet anomali olay ele alınmıştır ve bu dataset'te bulunan videolar frame'lerine ayrılarak etiketleme işlemi gerçekleştirilmiştir. Bu etiketlemelerden sonra ağa gönderilecek frame paketlerinin boyutunun belirlenmesi önem teşkil etmektedir. 3 adet frame paketi göz önüne alınmıştır. Bunlar 12, 16 ve 20 sayılarıdır. Ağdan elde edilen sonuçlar ve Sayın Prof. Dr. Murat EKİNCİ hocamızla yaptığımız değerlendirmeler sonucunda frame paket boyutu 20 olarak belirlenmiştir. Bu belirlemeden sonra ağ yapısını 20'lik paketlere göre güncelledik. Ayrıca *prediction* kodunda da bu açıdan değişikliğe gidilmiştir.

Yapılan çalışmalarda ağ yapısının ne kadar önemli olduğu konusunda farkındalık oluşmuştur. Hangi ağ yapısının nasıl performans verdiğinin ölçülmesi ve doğru olanda karar kılınması çok önemli bir etken olmaktadır. Bu nedenle birçok bilinen ağ modeli ve yöntemi denenmiştir. Üstelik özgün ağ tasarımları da yapılmıştır. Bu ağlardan elde edilen sonuçlar ve grafikler karşılaştırılmıştır. Bu karşılaştırmalar sonucunda hangi ağın kullanılacağına karar verilmiştir.

Bu kısma kadar dataset'te bulunan videoların dağılımları göz önüne pek alınmamıştır. Frame paket sayısına ve ağa karar verildikten sonra dataset'te bulunan sınıfların frame paket sayılarında eşitlemeye gidilmiştir. Bu eşitleme sonucunda ağın eşitlemeden önce elde edilen sonuçlara kıyasla daha iyi sonuçlar ürettiği gözlemlenmiştir. Frame paketlerinin eşitlenmesinin ne kadar önemli ve sonuca etki eden bir olay olduğunun farkına varılmıştır. Bu elde edilen sonuçlar ile ağ ve dataset son halini almıştır. Bu şekilde dataset'te olmayan bir video ile test işlemi gerçekleştirilmiştir. Bunun sonucunda %70 üstünde bir sonuç elde edilmiştir. Bu açıdan yapılan çalışmalar iyi bir şekilde sonuç vermiştir.



Şekil26- Elde edilen sonuç

4. Öneriler

Bu yapılan çalışmalar sonucunda tahmin etme başarısı %70 - %80 arasında olmaktadır. Tahmin sırasında oranı arttırmak amaçlı dataset'in genişletilmesi bu açıdan faydalı olacaktır. Dataset genişletilirken dikkat edilmesi gereken en önemli noktalardan biri var olan sınıfların frame paketlerinin eşit şekilde dağıtılmasını sağlamaktır. Ağın iyi bir performans vermesi için bu çok önemli bir durumdur. Eğer sınıf sayısı artırılmak isteniyorsa bu da kolay şekilde gerçekleştirilebilir. Ağın çıkış nöron sayısı artırılarak bu yapılabilir. Ayrıca dataset'teki dağılımlara dikkat ederek bu eklenecek yeni sınıfın frame paket sayısını ona göre ayarlamak gerekmektedir. Buna da dikkat edilmesi gerekmektedir.

Burada elde edilen ağdan daha iyi bir performans alınacağı düşünülen bir ağ olması durumunda bunun projeye uygulanarak gerekli testler yapılarak iyi veya kötü sonuç verdiği yorumlanabilir. Ağdan elde edilen grafikler ile bu kolay şekilde anlaşılabilir.

5. Kaynaklar

1. Proje kaynağı ve dataset: **https://www.crcv.ucf.edu/projects/real-world/**

2. Tensorflow kütüphanesi: **https://www.tensorflow.org/tutorials**

3. Keras: **https://keras.io/**

4. Dataset örnekleri: **https://www.kaggle.com/**

5. Bazı kod örnekleri : **https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/**

6. Visual Studio: **https://visualstudio.microsoft.com/tr/**

7. Python: **https://www.python.org/**

8. PyCharm: **https://www.jetbrains.com/pycharm/**

9. Github: **https://github.com/**

6. Ekler

Projede yapılmış çalışmalar ile ilgili kodlar, grafikler ve ağ mimarilerine bu linkten ulaşılabilir. Ayrıca herhangi bir sorun veya problem olması durumunda Onur ERDAŞ ve Emre ÖZDEMİR ile iletişime geçilebilir.

Link:

https://drive.google.com/file/d/1fAksfycFL2qQcz3YjOVx7RCGyTimxsIJ/view?usp=sharing

Demo Videosu:

https://drive.google.com/file/d/18LOGxLe8gnAbD6yIAvXrtE47jGI-gR1t/view

STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Var olan bir projenin tekrar yapılmasıdır. Farklı olarak C++'da ağırlıkları alınarak ağın kurulması kodlanmıştır.

2. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Hayır

3. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

Yapay Sinir Ağları ve programlama derslerinde edindiğimiz bilgileri kullandık.

4. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

Kod yazarken yazılım mühendisliği standartlarına uyarak kod yazımı yapıldı.

5. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

a) Ekonomi

Donanım üzerinde bu uygulamanın koşulması açısından daha az sistem gereksinimi sağlanmıştır.

b) Çevre sorunları:

Herhangi bir çevre sorunu teşkil etmemektedir.

c) Sürdürülebilirlik:

Dataset genişletilerek sürdürülebilirliği sağlanabilir.

d) Üretilebilirlik:

Fazla bir maliyet kalemi gerektirmemektedir.

e) Etik:

Etik açısından herhangi bir sıkıntı çıkartmamaktadır.

f) Sağlık:

Sağlık açısından bir sorun teşkil etmemektedir.

g) Güvenlik:

Güvenlik açısından bir sorun teşkil etmemektedir.

h) Sosyal ve politik sorunlar:

Sosyal ve politik bir sorun çıkartmamaktadır.