ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Mert ÇOPUR

MODIFIED RECURRENT CONVOLUTIONAL NEURAL NETWORKS FOR ACTION RECOGNITION

DEPARTMENT OF COMPUTER ENGINEERING

ADANA, 2019

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MODIFIED RECURRENT CONVOLUTIONAL NEURAL NETWORKS FOR ACTION RECOGNITION

Mert ÇOPUR

MSc THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on 18/07/2019.

…….....	……..……	……..….....
Asst. Prof. Dr. Buse Melis ÖZYILDIRIM	Prof. Dr. Selma Ayşe ÖZEL	Asst. Prof. Dr. Onur ÜLGEN
SUPERVISOR	MEMBER	MEMBER

This MSc Thesis is written at the Computer Engineering Department of Institute of Natural And Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

This work is supported by the Çukurova University Academic Research ProjectsUnit.
Project Number: FYL-2018-11163

ABSTRACT

MSc THESIS

MODIFIED RECURRENT CONVOLUTIONAL NEURAL NETWORKS FOR ACTION RECOGNITION

Mert ÇOPUR

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor : Asst. Prof. Dr. Buse Melis ÖZYILDIRIM
Year: 2019, Pages: 73
Jury :Prof. Dr. Selma Ayşe ÖZEL
:Asst. Prof. Dr. Onur ÜLGEN
:Asst. Prof. Dr. Buse Melis ÖZYILDIRIM

In this study, two new neural networks are proposed for action recognition. The first method is named as modified recurrent CNN and the second method is named as feature concatenating CNN. The main aim for proposing these networks is developing new techniques for sharing features between different frames of the videos. These methods have been applied on different datasets. Those datasets are UCF101, Hollywood2 and HMDB51. Based on the results, modified recurrent CNN is a good alternative for facilitating better feature learning. In most of the cases, accuracies provided by the modified recurrent neural networks are a few percent larger than accuracies provided by the standard convolutional neural networks. Additionally, a detailed review of the most important action recognition methods that are based on deep learning has been provided.

Keywords: Convolutional Neural Networks, Action Recognition, Recurrent Neural Networks, Feature Learning, Recurrent Convolutional Neural Network

ÖZ

YÜKSEK LİSANS TEZİ

MODİFİYE ÖZYİNELEMELİ KONVOLÜSYONEL SİNİR AĞLARI İLE HAREKET TANIMA

Mert ÇOPUR

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Dr. Öğr. Üyesi Buse Melis ÖZYILDIRIM
Yıl: 2019, Sayfa: 73
Jüri :Prof. Dr. Selma Ayşe ÖZEL
:Dr. Öğr. Üyesi Onur ÜLGEN
:Dr. Öğr. Üyesi Buse Melis ÖZYILDIRIM

Bu çalışmada hareket tanıma için iki yeni sinir ağı önerilmiştir. İlk yöntem modifiye özyinelemeli konvolüsyonel sinir ağı, ikinci yöntemse öznitelik birleştirmeli sinir ağı olarak isimlendirilmiştir. Bu yöntemlerin önerilme sebebi aynı videodaki çerçeveler arasında öznitelik paylaşımını sağlamak için yeni teknikler geliştirmektir. Bu yöntemler farklı veri setleri üzerinde uygulanmıştır. Bu veri setleri UCF101, HMDB51 ve Hollywood2'dir. Sonuçlara göre modifiye özyinelemeli konvolüsyonel sinir ağları daha iyi öznitelik öğrenmek için iyi bir alternatiftir. Modifiye özyinelemeli konvolüsyonel sinir ağları birçok durumda standart konvolüsyonel sinir ağlarından yüzde birkaç puan kadar daha iyi sonuç vermektedir. Ayrıca, derin öğrenmeye dayalı hareket tanıma tekniklerinin en önemlilerinin bir incelemesi sağlanmıştır.

Anahtar Kelimeler: Konvolüsyonel Sinir Ağları, Hareket Tanıma, Özyinelemeli Sinir Ağları, Öznitelik Öğrenme, Özyinelemeli Konvolüsyonel Sinir Ağları

EXTENDED ABSTRACT

Computer vision is a subfield of artificial intelligence which is interested in developing systems to understand visual information. It has a variety of applications such as autonomous product control systems in factories, drones used for surveillance and biometric authentication software. In recent years, deep learning based algorithms outperformed traditional algorithms. Therefore, traditional algorithms are replaced by deep learning based new algorithms. Two factors make it possible to benefit from full potential of deep learning. Those are availability of large datasets and usage of GPUs for general purpose computing. Deep learning algorithms require large amount of data to be trained on. We are generating huge size of data with the help of smart portable devices and other technologies. Therefore, more data is available for training and testing deep neural networks than before. GPUs are very suitable for parallel computation and common usage of them enables getting research output much faster.

Computer vision has many different areas and action recognition is one of the most important areas in computer vision. Action recognition defines the task of determining the category of the action carried out in a video. For example, we can develop a software to distinguish between videos of water sports and videos of soccer games.

Action recognition has very important applications in different areas. For example, visual surveillance systems are used for maintaining security. They detect suspicious actions relying on a stream of frames coming from a network of cameras and action recognition algorithms. Another application area for action recognition is video retrieval. With the widespread usage of internet, we produce of thousands of videos every day. Using only the associated text data for classification of videos may not result in the right predictions. Action recognition algorithms can provide essential information about video content.

Action recognition algorithms rely on two different types of information. Those are spatial information and temporal information. A single frame in a video can provide lots of information about the objects and the environments involved in the

action. The information that can be provided by a single frame is called as spatial information. A video is a collection of frames. Movements of individuals, objects and environment in a video form an action. This can be detected when information from consecutive frames are taken into account. This type of information is called as temporal information. An image classifier can be used to classify frames by taking into account the spatial information. A prediction can be made for whole video by classifying frames. However, most of the action recognition algorithms benefit from both spatial information and temporal information.

In this thesis, two new approaches are suggested for action recognition. The first approach is named as modified recurrent convolutional neural network and the second approach is named as feature concatenating convolutional neural network.

In the proposed modified recurrent convolutional neural network, each recurrent convolutional layer (RCL) receives the output of the same layer for the previous frame and the output of the previous layer of the same frame. Recurrent convolution layers receive two different inputs. Those are feed-forward input and recurrent input. The output of the previous layer is the feed-forward input. This is the only input in the standard convolutional layer. The output of the same layer for the previous frame is the recurrent input. Convolution operation is applied for recurrent input and feed-forward input with separate weights and biases. The result of convolution on feed-forward input is multiplied with 0.7 and the result of convolution on recurrent input is multiplied with 0.3. After that, the result of two multiplications are added to find the output of the recurrent convolutional layer.

The main aim of feature concatenating CNN is same with modified recurrent convolutional neural networks. They both incorporate information from previous frames. However, feature concatenating CNN is not as sophisticated as modified recurrent CNN. In feature concatenating CNN, each layer has only 1 convolution operation instead of 2. All convolutional layers except the first layer receive two inputs. Those are the output of previous layer and the output of previous layer for the previous frame. Those outputs are concatenated.

We have used 3 datasets for training and testing. Those are Hollywood2, HMDB51 and UCF101. We have created 9 neural networks. 3 of them receive a single frame as input. 3 of them receive optical flow displacement fields. 3 of them receive a volume of frames to use 3D convolution operation. Networks receiving same type of input form a category. In each category, there is a standard convolutional network, a modified recurrent network and a feature concatenating network. Additionally, two-stream networks have been tested for each category. The obtained results indicate that the modified recurrent CNN is useful alternative to other methods. It can outperform standard convolutional networks in a lot of cases. However, feature concatenating convolutional neural networks fall behind standard convolutional networks most of the time. The performances depend on the dataset and the category of the network. In most of the cases, accuracies provided by the modified recurrent neural networks are a few percent larger than accuracies provided by the standard convolutional neural networks.

The output of the two-stream CNN is calculated by averaging softmax scores of the single-frame CNN and the optical flow CNN. The output of the two-stream modified recurrent CNN is calculated by averaging softmax scores of the modified recurrent CNN and the optical flow modified recurrent CNN. The output of the two-stream feature concatenating network is calculated by averaging the feature concatenating CNN and the optical flow feature concatenating CNN. Additionally, baseline models were generated for all datasets by randomly generating predictions for all videos. The accuracy of the baseline model is the percentage of the videos that are predicted correctly.

During training and testing, 20 frames have been used from each video. Therefore, 20 frames have been sampled from each video. There is a constant number of frames between two sampled frames and it is called as step size. Step size is calculating by dividing the total number of frames by 20. For example, if a video has 40 frames, the step size will be 2 because 20 frames will be sampled from 40 frames. The first frame of the video will be the first sampled frame. The third frame of the video will be the second sampled frame. The fifth frame of the video will be the third sampled frame. Videos having less than 20 frames are discarded.

Only one frame is inputted to the networks at a time. Frames that are sampled from the same video are inputted to the networks consecutively. When a frame is inputted to the network, the gradients are calculated. Gradients for all frames that belongs to the same video is summed. Then, the training algorithm is run using the sum of the gradients. This training strategy is used because it is suitable for modified recurrent CNN and feature concatenating CNN. With this training strategy, feature maps can be transmitted between sampled frames easily which is required by modified recurrent CNN and feature concatenating CNN. A different training strategy has also been investigated. Each time a number of frames with the same order have been sampled from different videos. For example, if batch size is 20, the first sampled frames from 20 different videos form the batch for the first epoch. For the second epoch, the second sampled frames are used. This approach did not give satisfying results in the experiments. Because, in each epoch, the network learns to classify frames at a specific position. When training is complete, the network can only classify frames that are located at the end of the video.

The 20 sampled frames are classified separately in all videos. Based on predictions of the frames, predictions for the videos are calculated. We report two different accuracies for each method. The first metric of accuracy is overall accuracy. Overall accuracy is calculated by taking the most common prediction among the sampled frames in a video as the prediction for a video. The second metric of accuracy is accuracy based on the last frame. It is calculated by taking the prediction of the last sampled frame in a video as the prediction for a video. Discussions on the results are held on overall accuracies because in action recognition studies, the most common metric for evaluation is video accuracy.

For future work, larger networks and better testing strategies can be used to obtain better performance. Currently, our networks have less filters and less layers than those of the studies that obtain competitive performances. Secondly, alternative ways can be explored to incorporate features from different frames.

GENİŞLETİLMİŞ ÖZET

Yapay zekanın bir alt alanı olan yapay görme görsel bilgiyi anlayan sistemler geliştirmeyle ilgilenmektedir. Fabrikalardaki otonom kontrol sistemleri, güvenlik için kullanılan dronelar ve biyometrik kimlik doğrulama sistemleri gibi birçok uygulama alanı vardır. Son yıllarda, derin öğrenmeye dayalı yöntemler geleneksel yöntemleri geçmektedir. Bu sebeple, geleneksel yöntemlerin yerine derin öğrenmeye dayalı yöntemler kullanılmaktadır. Derin öğrenmenin tüm potansiyelinin kullanılmasını mümkün kılan iki neden vardır. Bunlar büyük veri setlerinin erişilebilir olması ve GPU'ların genel amaçlı hesaplamalar için kullanılmasıdır. Derin öğrenmeye dayalı algoritmaların eğitilmesi için çok fazla veri gerekmektedir. Akıllı taşınabilir cihazların ve diğer teknolojilerin yardımı ile devasa miktarda veri üretmekteyiz. Bu nedenle, derin sinir ağlarının eğitimi ve testi için eskisinden daha fazla veri mevcuttur. GPU'lar paralel hesaplama için çok uygundur ve yaygın kullanımları sayesinde araştırma sonucuna daha hızlı ulaşılabilmektedir.

Yapay görmenin birçok alanı vardır ve hareket tanıma bunların en önemlilerinden biridir. Hareket tanıma, bir videodaki hareketin hangi kategoriye ait olduğuna karar verme işini tanımlar. Örneğin, su sporlarının videoları ile futbol maçlarının videolarını birbirinden ayırmak için bir yazılım geliştirilebilir.

Hareket tanımanın farklı alanlarda birçok uygulaması vardır. Örneğin, görüntülü güvenlik sistemleri güvenliğin sağlanması için kullanılmaktadır. Bir kamera ağından gelen görüntüler ve hareket tanıma algoritmaları kullanılarak şüpheli durumlar tespit edilir. Hareket tanımanın bir diğer uygulama alanı ise video verisi çekmedir. İnternetin yaygın kullanımı ile birlikte her gün binlerce video üretilmektedir. Yalnızca videonun metnini kullanarak sınıflama yapmak doğru tahminler sağlamayabilir. Hareket tanıma algoritmaları videonun içeriği hakkında önemli bilgi sağlayabilir.

Hareket tanıma algoritmaları iki tür bilgi kullanır. Bunlar uzaysal bilgi ve zamansal bilgidir. Videodaki tek bir kare harekete dahil olan nesneler ve ortamlar hakkında çok fazla bilgi verebilir. Tek bir kare tarafından sağlanabilecek bu tür bilgilere uzaysal bilgi denir. Bir video çeşitli karelerden oluşur. Videodaki kişilerin,

nesnelerin ve ortamın eylemleri hareketi meydana getirir. Bu ancak ardışık karelerden gelen bilgiler kullanılarak tespit edilebilir. Bu tür bilgilere zamansal bilgi denir. Yalnızca uzaysal bilgiyi kullanan bir görüntü sınıflayıcı videodaki kareleri sınıflandırmak için kullanılabilir. Kareleri sınıflandırarak videonun sınıfı tahmin edilebilir. Bununla birlikte, birçok hareket tanıma algoritması hem uzaysal hem zamansal bilgiden yararlanmaktadır.

Bu tezde hareket tanıma için iki yeni yaklaşım önerilmiştir. Bunlardan birincisi modifiye özyinelemeli konvolüsyonel sinir ağları, ikincisi ise öznitelik birleştirmeli konvolüsyonel sinir ağları olarak isimlendirilmiştir.

Önerilen modifiye özyinelemeli konvolüsyonel sinir ağında her bir özyinelemeli konvolüsyonel katman bir önceki katmanın ve bir önceki kare için aynı katmanın çıkış değerlerini alır. Özyinelemeli konvolüsyonel katmanlar iki çeşit girdi alır. Bunlar ileri beslemeli girdi ve özyinelemeli girdi olarak isimlendirilir. Bir önceki katmanın çıkışı ileri beslemeli girdidir. Standart konvolüsyonel katmandaki tek girdi ileri beslemeli girdidir. Bir önceki kare için aynı katmanın çıkışı ise özyinelemeli girdidir. Konvolüsyon işlemi özyinelemeli ve ileri girdiler için ayrı ağırlık ve bias değerleri ile yapılır. İleri beslemeli girdi üzerine uygulanan konvolüsyon işleminin sonucu 0.7 ile, özyinelemeli girdi üzerine uygulanan konvolüsyon işleminin sonucu ise 0.3 ile çarpılır. Ardından iki çarpımın sonucu toplanarak özyinelemeli konvolüsyonel katmanın sonucuna ulaşılır.

Öznitelik birleştirmeli konvolüsyonel sinir ağlarının amacı modifiye özyinelemeli konvolüsyonel sinir ağları ile aynıdır. Her ikisi de önceki karedeki bilgiden yararlanmaktadır. Bununla birlikte, öznitelik birleştirmeli konvolüsyonel sinir ağları modifiye konvolüsyonel sinir ağları kadar sofistike değildir. Öznitelik birleştirmeli konvolüsyonel sinir ağlarında her katmanda iki yerine yalnızca bir konvolüsyon işlemi bulunmaktadır. İlk katman dışındaki bütün katmanlar iki tane girdi almaktadır. Bunlar bir önceki katmanın çıktısı ve bir önceki katmanın bir önceki kare için ürettiği çıktıdır. Bu çıktılar birleştirilir.

Eğitim ve test için 3 farklı veri seti kullanılmıştır. Bunlar Hollywood2, HMDB51 ve UCF101 veri setleridir. 9 farklı yapay sinir ağı geliştirilmiştir. Bunlardan

3 tanesi girdi olarak tek bir kare alırlar. Üç tanesi ise optik akış değerleri alırlar. Diğer üç tanesi ise 3 boyutlu konvolüsyon işlemi ile kullanmak için birden fazla kare alırlar. Aynı tür girdiyi alan ağlar bir kategori meydana getirirler. Her bir kategoride bir standart konvolüsyonel ağ, bir modifiye özyinelemeli konvolüsyonel ağ ve bir öznitelik birleştirmeli ağ bulunmaktadır. Ayrıca her bir kategori için iki-akışlı ağlar test edilmiştir. Ulaşılan sonuçlar modifiye özyinelemeli konvolüsyonel sinir ağlarının diğer yöntemlere iyi bir alternatif olduğunu ortaya koymaktadır. Modifiye özyinelemeli konvolüsyonel sinir ağları birçok durumda standart konvolüsyonel ağları geride bırakmaktadır. Öznitelik birleştirmeli yapay sinir ağları ise birçok durumda standart konvolüsyonel ağların gerisinde kalmaktadır. Performanslar veri setine ve ağın kategorisine bağlıdır. Birçok durumda, modifiye özyinelemeli konvolüsyonel ağlar tarafından sağlanan performans standart konvolüsyonel ağlar tarafından sağlanan performansın yüzde birkaç puan üstündedir.

İki akışlı konvolüsyonel sinir ağının sonucu standart konvolüsyonel ağın ve optik akışlı standart konvolüsyonel ağın softmax sonuçlarının ortalaması alınarak bulunur. İki akışlı modifiye özyinelemeli konvolüsyonel sinir ağının sonucu modifiye özyinelemeli konvolüsyonel ağın ve optik akışlı modifiye özyinelemeli konvolüsyonel ağın softmax sonuçlarının ortalaması alınarak bulunur. İki akışlı öznitelik birleştirmeli konvolüsyonel sinir ağının sonucu öznitelik birleştirmeli konvolüsyonel ağın ve optik akışlı öznitelik birleştirmeli konvolüsyonel ağın softmax sonuçlarının ortalaması alınarak bulunur. Ayrıca, bütün veri setleri için videolara rastgele tahminler üreterek referans modeller oluşturulmuştur. Referans modellerin doğruluğu ise doğru tahmin edilen videoların yüzdesidir.

Eğitim ve test için her videodan 20 kare kullanılmıştır. Bu nedenle, her videodan 20 kare örneklenmiştir. Örneklenen iki kare arasında sabit sayıda kare vardır ve bu karelerin sayısına adım büyüklüğü denir. Adım büyüklüğü toplam kare sayısını 20 ile bölerek bulunur. Örneğin, bir videoda 40 kare varsa 20 kare örnekleneceği için adım büyüklüğü 2 olacaktır. Videodaki ilk kare ilk örneklenen kare olacaktır. Üçüncü kare videoda ikinci örneklenen kare olacaktır. Beşinci kare ise üçüncü örneklenen kare olacaktır. 20 kareden az kare içeren videolar kullanılmamıştır.

Ağlara tek seferde yalnızca bir kare girdi olarak verilmektedir. Aynı videodan örneklenmiş olan kareler ağa ardışık bir şekilde girdi olarak verilmektedir. Ağ bir kareyi aldıktan sonra türevleri hesaplar. Aynı videoya ait tüm kareler için hesaplanan türevler toplanır. Ardından bu toplam türev kullanılarak eğitim algoritması çalıştırılır. Bu eğitim stratejisinin kullanılma sebebi modifiye özyinelemeli konvolüsyonel sinir ağları ve öznitelik birleştirmeli konvolüsyonel sinir ağları için uygun olmasıdır. Modifiye özyinelemeli konvolüsyonel yapay sinir ağları ve öznitelik birleştirmeli konvolüsyonel sinir ağları örneklenen kareler arasında öznitelik haritalarının iletilmesini gerektirmektedir. Bu eğitim stratejisi sayesinde örneklenen kareler arasında öznitelik haritaları iletimi kolayca sağlanabilir. Farklı bir eğitim stratejisi de ayrıca denenmiştir. Farklı videolardan belli sayıda kare aynı sıradan olacak şekilde örneklenir. Örneğin, eğer yığın (batch) büyüklüğü 20 ise ilk devir (epoch) için 20 farklı videonun ilk örneklenen kareleri kullanılır. İkinci devir (epoch) için ise ikinci örneklenen kareler kullanılır. Yapılan deneylerde bu yaklaşım yeterli sonuç vermemiştir. Çünkü eğitilen ağ her devirde (epoch) belli bir konumdaki kareleri sınıflandırmayı öğrenmektedir. Eğitim tamamlandığında ağ yalnızca videonun sonundaki kareleri sınıflandırmayı öğrenmektedir.

Tüm videolarda örneklenen 20 kare ayrıca sınıflandırılmaktadır. Kareler için üretilen tahminlere dayanarak videolar için tahminler hesaplanmaktadır. Her yöntem için iki farklı doğruluk metriği kullanılmıştır. Bunlardan birincisi ortalama doğruluktur. Ortalama doğruluk bir videodan örneklenen kareler için üretilen sınıf tahminleri arasında en yaygın olanın videonun sınıfı olarak alınmasıyla hesaplanır. İkinci doğruluk metriği ise son kareye dayalı doğruluktur. Bir videodan örneklenen son kare için tahmin edilen sınıf videonun sınıfı olarak kabul edilir. Sonuçlar üzerine yapılan tartışmalar ortalama doğruluğa dayanılarak yapılmıştır çünkü hareket tanıma çalışmalarında kullanılan en yaygın metrik video doğruluğudur.

İleride yapılacak çalışmalarda, daha büyük ağlar ve daha iyi test stratejileri kullanılarak daha iyi sonuçlara ulaşılabilir. Kullandığımız ağlar en iyi sonuçlara ulaşan yöntemlerde ki ağlara göre daha az filtre ve katman içermektedir. İkinci olarak, farklı karelerden gelen özniteliklerden yararlanmak için alternatif yöntemler araştırılabilir.

ACKNOWLEDGMENT

I would like to express my endless thanks and appreciation to my supervisor, Asst. Prof. Dr. Buse Melis Özyıldırım, for all respectable knowledge, support, and patience. Finally, I want to thank my family for always supporting me.

CONTENTS **PAGES**

LIST OF FIGURES **PAGES**

LIST OF TABLES **PAGES**

ABBREVIATIONS

CNN	: Convolutional Neural Networks
MLP	: Multi Layer Perceptron
RNN	: Recurrent Neural Network
SRN	: Simple Recurrent Neural Network
LSTM	: Long-Short Term Memory Recurrent Neural Network
RCNN	: Recurrent Convolutional Neural Network
RCL	: Recurrent Convolutional Layer
ARTNet	: Appearance-and-Relation Networks
SMART	: Simultaneously model appearance and relation

1. INTRODUCTION

Computer vision is a subfield of artificial intelligence which is interested in developing systems to understand visual information. It has a variety of applications such as autonomous product control systems in factories, drones used for surveillance and biometric authentication software. In recent years, deep learning based algorithms outperformed traditional algorithms. Therefore, traditional algorithms are replaced by deep learning based new algorithms. Two factors make it possible to benefit from full potential of deep learning. Those are availability of large datasets and usage of GPUs for general purpose computing. Deep learning algorithms require large amount of data to be trained on. We are generating huge size of data with the help of smart portable devices and other technologies. Therefore, more data is available for training and testing deep neural networks than before. GPUs are very suitable for parallel computation and common usage of them enables getting research output much faster.

Computer vision has many different areas and action recognition is one of the most important areas in computer vision. Action recognition defines the task of determining the category of the action carried out in a video. For example, we can develop a software to distinguish between videos of water sports and videos of soccer games.

There are two subfields of computer vision that are related to action recognition. The first one is action prediction. The main difference between action recognition and action prediction is that the actions have to be predicted before they are carried out in the video in action prediction while the prediction happens after the action is carried out in the action recognition. Action prediction is useful in cases in which the software has to react before the action (Kong & Fu, 2018). In action recognition, the main goal is assigning the right class to a video. There is a different area in computer vision named as action localization. In action

localization, the goal is to both classify the action and localize the action in space and time (Kalogeiton, Weinzaepfel, Ferrari, & Schmid, 2017).

Action recognition has very important applications in different areas. For example, visual surveillance systems are used for maintaining security. They detect suspicious actions relying on a stream of frames coming from a network of cameras and action recognition algorithms. Another application area for action recognition is video retrieval. With the widespread usage of internet, we produce of thousands of videos every day. Using only the associated text data for classification of videos may not result in the right predictions. Action recognition algorithms can provide essential information about video content (Kong & Fu, 2018).

Video gaming is a very popular way of entertainment. Some games let us interact with the virtual world without using any physical device like keyboard or gamepad. In this type of games, the user uses an RGB-D camera like Kinect to interact with the game with their body movements. Action recognition algorithms are important parts of these systems. Additionally, action recognition algorithms can increase the level of interaction between humans and robots when robots are equipped with them (Kong & Fu, 2018).

Action recognition algorithms rely on two different types of information. Those are spatial information and temporal information. A single frame in a video can provide lots of information about the objects and the environments involved in the action. The information that can be provided by a single frame is called as spatial information. A video is a collection of frames. Movements of individuals, objects and environment in a video form an action. This can be detected when information from consecutive frames are taken into account. This type of information is called as temporal information. An image classifier can be used to classify frames by taking into account the spatial information. A prediction can be made for whole video by classifying frames. However, most of the action recognition algorithms benefit from both spatial information and temporal information.

There are many factors which make action recognition a difficult area to study. Firstly, samples belonging to the same class can have very big variations among them. For example, running action can be done differently by each person or each video can be taken from a different camera position. Furthermore, different action classes can be expected to be similar in terms of their motion patterns. For instance, walking and running actions have similar motion patterns (Kong & Fu, 2018).

Another factor is diversity of backgrounds and camera motion. Most of the videos are taken in outdoor environments where the background is dynamic and the lighting conditions are unpredictable. The location of the camera is not constant in most of the videos which further increases the difficulty of the task. Training a deep neural network with millions of parameters requires utilization of very large datasets. Although datasets like UCF-101 and HMDB51 contain thousands of videos, they are not sufficient. We have very large action recognition datasets like Sports-1M and Youtube-8M. However, they were collected by video retrieval methods and may not be completely accurate. The last obstacle of action recognition to mention is the uneven predictability of frames. Each video consists of many frames. However, not every frame contains information about the class of the video. Furthermore, some frames may include an action of another class. For example, a video which contains talking people can also include other people who are walking (Kong & Fu, 2018).

1.1. Main Objective

To recognize the action in a video, an action recognition algorithm has to learn the relations among frames. Different algorithms achieve this by different ways. For example, some algorithms process more than one frame at once and some algorithms use optical flow values as input. Action recognition is a very popular research area and new papers present different ways to the same problem.

In this thesis, two related new approaches will be suggested for learning relations between frames in the video.

1.2. Thesis Contribution

In this thesis, two new approaches are suggested. The first approach is similar to a recurrent convolutional neural network. However, it is implemented between frames. In this approach, each convolutional layer receives two inputs. The first input is the feed-forward input which comes from the previous layer. The second input is the recurrent input which is the output of the same layer for the previous frame. These two inputs are used together to form an output. The second approach is similar to the first approach. The main difference between them is that in the second approach, the inputs are concatenated instead of doing calculations using both of them to form an output. Both methods will be explained in more detail in the third section.

The accuracy of the methods was evaluated by experimentation. Deep convolutional neural networks are very successful and popular for deep learning tasks. In most of the areas of the computer vision, the state-of-the-art method is based on convolutional neural networks. Therefore, the two approaches are applied on some selected methods which are all based on convolutional neural networks. The selected methods are discussed in detail in Chapter 2. In the Chapter 4, all results are presented to be able to see the effects of the suggested approaches experimentally.

1.3. Thesis Layout

The content of the following sections is organized as follows. Chapter 2 contains background information and some related studies including the methods which are applied in the thesis are presented. In Chapter 3, the datasets, tools and network configurations are presented. Also, all the used parameters are defined there. Chapter 4 is the section which results are reported and they are discussed.

Chapter 5 is the conclusion section which a short summary of the thesis is included. Also, potential future research related to the thesis is discussed in the conclusion.

2. BACKGROUND

2.1. Convolutional Neural Networks

In recent years, convolutional neural networks (CNN) achieved very successful results in various computer vision problems. Thus, more and more researchers are using CNNs. For my research, CNNs will be the base algorithm that I am building on. To understand CNNs, multilayer perceptrons (MLP) (Werbos, 1974) should be analyzed firstly because MLPs form the basis of CNNs. An MLP consists of a number of layers where each layer has a number of computational units called neurons. A neuron computes its output by subtracting a bias from weighted sum of its input and applying a non-linear function to the result. Figure 2.1 illustrates the structure of a neuron. The equation calculated by a neuron for computing the output is given below.

$$o_j = \phi\left(\sum_i w_{ij}\, x_i - \Theta_j\right) \tag{2.1}$$

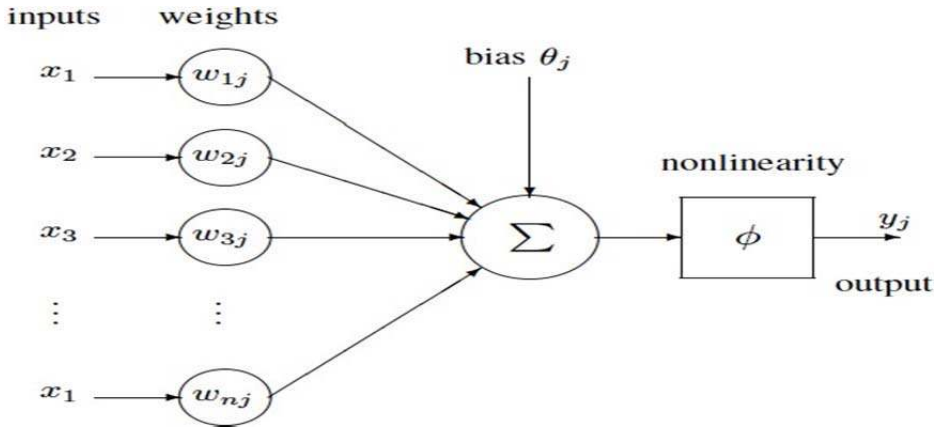

Figure 2.1. Structure of a neuron (Castelluccio, Poggi, Sansone, & Verdoliva, 2015)

In MLP, weights are learnt by training to solve a particular problem using representative training samples. The most common training algorithm is backpropagation (Werbos, 1990). In MLPs, each neuron in a layer is connected to all neurons in the following layer. The main problem in this structure is huge number of parameters to be learned. CNNs overcome this issue in two ways. Firstly, neurons in CNNs are not connected to all neurons in the following layer. Secondly, a set of weights is shared throughout the layer. Figure 2.1 illustrates the difference between a layer of an MLP and a layer of a CNN. (Castelluccio, Poggi, Sansone, & Verdoliva, 2015)

Typically, convolutional layers are followed by pooling layers. The task of pooling layer is subsampling the data using non-linear operations like max(). After a series of convolutional and pooling layers, fully-connected layers are added to the network. Fully connected layers are simply densely connected layers as seen in MLPs (Castelluccio, Poggi, Sansone, & Verdoliva, 2015).

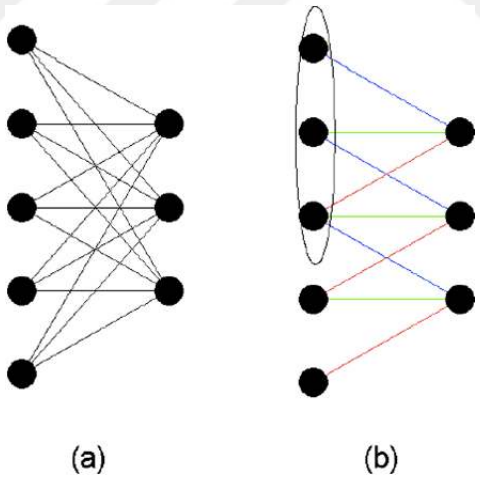

Figure 2.2. In the MLP layer (a), there is a separate connection between each pair of neurons where each connection has a separate weight. In the CNN layer (b), each neuron is connected to a limited number of neurons. Also, connections with same color share the weights. (Castelluccio, Poggi, Sansone, & Verdoliva, 2015)

2.2. Recurrent Neural Networks

Recurrent neural networks (RNNs) are used for learning dependencies in sequential data. There are two common types of recurrent neural networks. The first one is simple recurrent neural network (SRN) and the second one is long-short term memory recurrent neural network (LSTM). In the following equations, $x^{(t)}$, $h^{(t)}$ and $y^{(t)}$ represent input, hidden and output values in the t-th time state (Zuo, Shuai, Wang, Liu, Wang, & Wang, 2016).

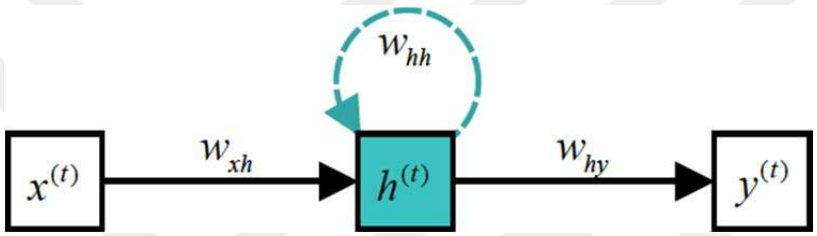

Figure 2.3. Structure of SRN (Zuo, Shuai, Wang, Liu, Wang, & Wang, 2016).

As illustrated in the Figure 2.3, calculating hidden state in a time state *t* requires hidden state of previous time state and input state at the current time state. Thus, $h^{(t)}$ and $y^{(t)}$ in a certain time step can be calculated as:

$$h^{(t)} = \psi_h\left(W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h\right) \tag{2.2}$$

$$y^{(t)} = \psi_y\left(W_{hy}h^{(t)} + b_y\right) \tag{2.3}$$

In the equations, W_{hh}, W_{xh} and W_{hy} are parameters to be learned. The ψ_h and ψ_y are non-linear activation functions. SRNs are not successful at learning dependencies in long sequences. Thus, LSTMs were developed (Zuo, Shuai, Wang, Liu, Wang, & Wang, 2016).

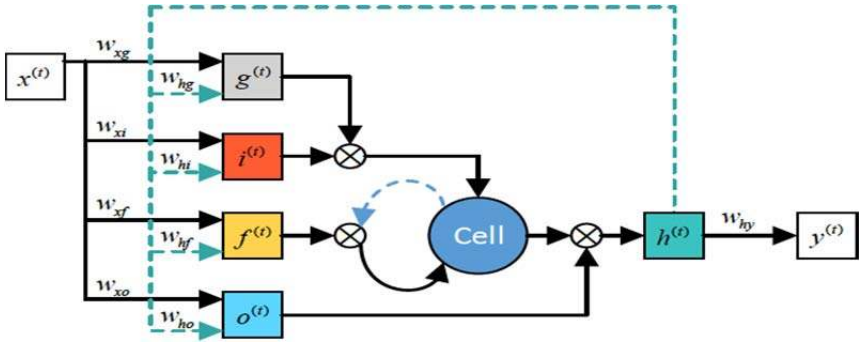

Figure 2.4. Structure of LSTM (Zuo, Shuai, Wang, Liu, Wang, & Wang, 2016)

LSTM introduced a cell state *c(t)* and four gates whose names are input gate *i(t)*, output gate *o(t)*, forget gate *f(t)* and input modulation gate *g(t)*. At the t-th state, LSTM computes a series of equations as shown below.

$$i^{(t)} = \sigma\left(W_{hi}h^{(t-1)} + W_{xi}x^{(t)} + b_i\right) \tag{2.4}$$

$$f^{(t)} = \sigma\left(W_{hf}h^{(t-1)} + W_{xf}x^{(t)} + b_f\right) \tag{2.5}$$

$$o^{(t)} = \sigma\left(W_{ho}h^{(t-1)} + W_{xo}x^{(t)} + b_o\right) \tag{2.6}$$

$$g^{(t)} = \phi\left(W_{hg}h^{(t-1)} + W_{xg}x^{(t)} + b_g\right) \tag{2.7}$$

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)} \tag{2.8}$$

$$h^{(t)} = o^{(t)} \odot \phi\left(c^{(t)}\right) \tag{2.9}$$

$$y^{(t)} = \psi\left(W_{hy}h^{(t)} + b_y\right) \tag{2.10}$$

In the equations above, σ represents logistic sigmoid function, ϕ represents hyperbolic tangent nonlinearity and $\odot$ represents element-wise multiplication (Zuo, Shuai, Wang, Liu, Wang, & Wang, 2016).

2.3. Recurrent Convolutional Neural Networks

Recurrent convolutional neural networks include Recurrent Convolutional Layers (RCLs). An RCL can be unfolded to *T* time steps as seen in Figure 2.5. For the first time step, only feed-forward input exists. For other time steps, both feed forward input and the output of previous time step (recurrent input) are taken as input and used for calculation. Consider location *(i, j)* on the *kth* feature map. The output z_{ijk} is calculated as the following.

$$z_{ijk}(t) = \left(w_k^f\right)^T u^{(i,j)}(t) + (w_k^r)^T x^{(i,j)}(t-1) + b_k \tag{2.11}$$

In the equation above, the variables $u^{(i,j)}(t)$ and $x^{(i,j)}(t-1)$ represent the vectorized feed-forward input and the vectorized recurrent input, respectively. When *t* is 0, $x^{(i,j)}(t-1)$ is equal to 0. Thus, when *T* is set to 0, an RCL transforms to a convolutional layer. The $\left(w_k^f\right)$, (w_k^r) and b_k are vectorized feed-forward weight, vectorized recurrent weight and bias respectively. Since, weights and biases are shared among different time steps, the number of parameters to train does not become huge (Liang & Hu, 2015).

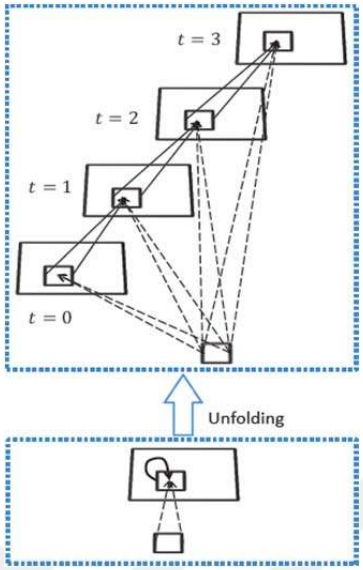

Figure 2.5. Unfolding an RCL for 3 times (Liang & Hu, 2015)

2.4. Action Recognition Algorithms

2.4.1. Extending CNNs for Action Recognition

The biggest difference between action recognition and image classification is that videos consist of both spatial and temporal components. Therefore, an action classification algorithm has to learn both spatial and temporal relations in the video to successfully classify videos. Several different approaches to extend CNNs to be applied for the task of action recognition were suggested by (Karpathy et al., 2014). Each approach has its own way to detect spatial and temporal relations.

2.4.1.1. Single-Frame CNN

A single video consists of many frames. A single-frame CNN treats action recognition problem as an image classification problem. Each frame is a static image. A single-frame CNN detects actions from the pixel values of frames. It is trained on frames from the videos. A single-frame CNN does not take into account temporal relations among frames in an image. However, it can still provide competitive results because appearance provides a lot of clues about the action.

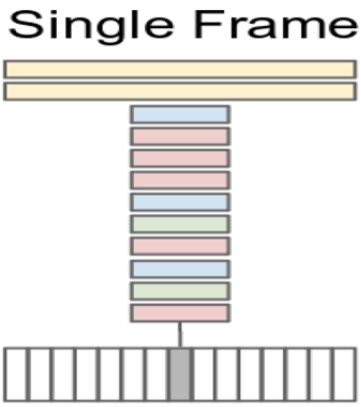

Figure 2.6. An illustration of a single-frame CNN. Red, green, blue and yellow boxes indicate convolutional, normalization, pooling and fully-connected layers respectively. (Karpathy et al., 2014)

2.4.1.2. Early Fusion CNN

For Early Fusion, consecutive pixels in a time-window are combined and fed into a network for classification. The first convolutional layer is extended to include temporal component.

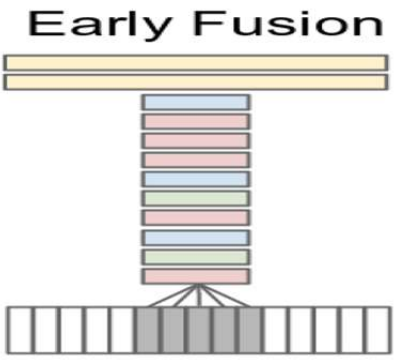

Figure 2.7. An illustration of Early Fusion CNN. Red, green, blue and yellow boxes indicate convolutional, normalization, pooling and fully-connected layers respectively. (Karpathy et al., 2014)

2.4.1.3. Late Fusion CNN

In the Late Fusion CNN, two single-frame networks are placed with a specified number of pixels between them. The two networks share parameters. Their outputs are combined in fully-connected layers. Two networks are not able to detect motion information. However, fully-connected layers can detect them based on combined output of networks.

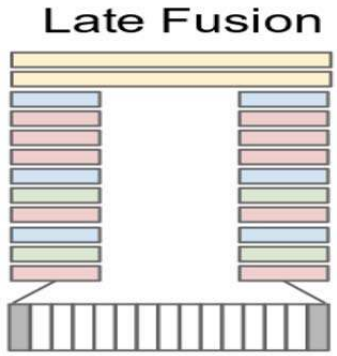

Figure 2.8. An illustration of Late Fusion CNN. Red, green, blue and yellow boxes indicate convolutional, normalization, pooling and fully-connected layers respectively. (Karpathy et al., 2014)

2.4.1.4. Slow Fusion CNN

Slow Fusion approach is a combination of Early and Late Fusion approaches. Each layer of the network consists of a few layers and each network is fed with a different time window. Networks in upper layers operates on the combination of outputs from the networks in the previous layer. All convolutional layers are extended to include temporal component in this type of networks. All networks in the same layer have to share parameters.

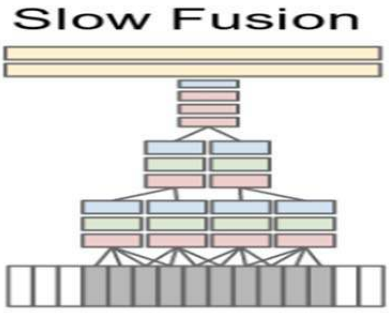

Figure 2.9. An illustration of Slow Fusion CNN. Red, green, blue and yellow boxes indicate convolutional, normalization, pooling and fully-connected layers respectively. (Karpathy et al., 2014)

2.4.2. Multiresolution CNNs

Training and testing can take very long amount of time in deep neural networks. One way to speed up the network is to decrease the size of the inputs. However, images with lower resolution contain less amount of details which causes lower accuracy. The main aim of Multiresolution CNN is to decrease running time without sacrificing accuracy. (Karpathy et al., 2014)

A Multiresolution CNN consists of two streams. Those are fovea stream and context stream. A center crop is taken from the image and inputted to fovea stream. The size of the crop is half of the original image. Usually photographer adjust the camera to make object of interest appear at the center of the image. Therefore, using the center crop is reasonable. For context stream, the resolution of the original image is decreased to half. The outputs of the both streams are combined and inputted to the first fully-connected layer.

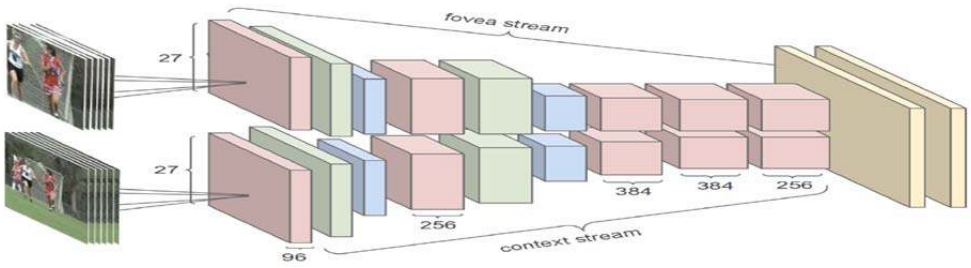

Figure 2.10. An illustration of Multiresolution CNN (Karpathy et al., 2014)

2.4.3. Two-Stream Convolutional Neural Networks

In the previously explained architectures, networks have to learn temporal relations based on pixel values. Pixel values cannot explicitly provide information about motion. Therefore, the networks have to implicitly extract motion patterns.

Two-Stream Convolutional Neural Networks have two streams. Those are named as spatial stream and temporal stream. Both of them are implemented as CNNs. The input of spatial stream is pixel values of frames. It is the same as single-frame CNN. It classifies videos based on their pixel values. Spatial stream can provide good results even when it is used alone because appearance of the frames in a video contains important clues for the action of the video. For example, videos of the same class can include the same type of object. Since spatial stream is basically an image classifier, image classification datasets can be used to pretrain the spatial stream. Temporal stream receives optical flow displacement fields between several consecutive frames as input. This type of input explicitly represents motion patterns between frames. The output of both streams is the softmax scores. The output of the two-stream network can be calculated in two ways. The softmax scores of the two streams can be averaged or the concatenated L2-normalized softmax scores can be trained on a multi-class linear SVM. (Simonyan & Zisserman, 2014)

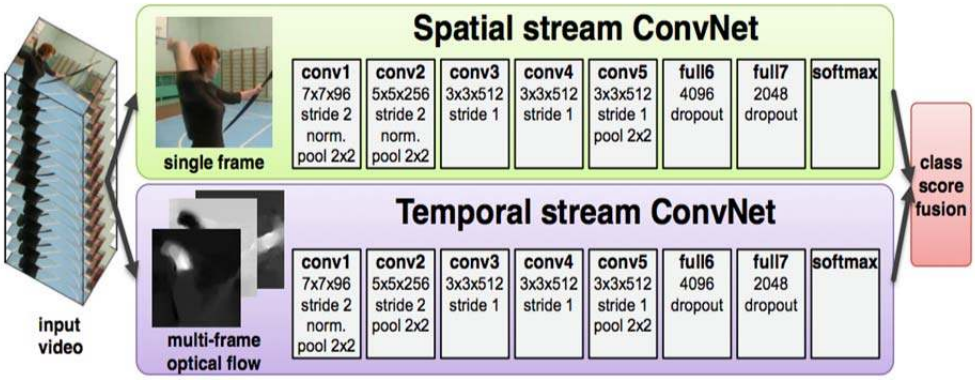

Figure 2.11. An illustration of a sample Two-Stream CNN (Simonyan & Zisserman, 2014)

2.4.4. Multi-task Learning

Video classification datasets are relatively smaller than image classification datasets. Therefore, deep neural networks suffer from overfitting. To prevent overfitting, usage of more training data is needed. The most straightforward way to increase number of training data is to combine multiple datasets. However, this is not a good approach because of two reasons. Firstly, some classes in the datasets may overlap. Combining all classes can result in duplicate training samples. Removing duplicate samples or repeating classes is not a trivial task. Secondly, a softmax has to be trained for multiple classes.

Multi-task learning provides a more sophisticated method for combining different datasets. The main goal is to train a neural network to be able to classify multiple datasets. For this purpose, a different softmax layer is added for each dataset after the last fully-connected layer. Each layer can have a different loss function. Each layer is responsible for only the respective dataset. The overall training loss is computed as the sum of all losses. (Simonyan & Zisserman, 2014)

2.4.5. 3D Convolutional Neural Networks

3D CNNs are more suitable to action recognition than 2D CNNs. In 3D CNNs, convolution and pooling operations are done in 3D. Therefore, they are able to process the data spatio-temporally. On the other hand, 2D convolution and 2D pooling operations can only process data spatially. The difference between 2D convolution and 3D convolution is clearly illustrated in Figure 2.12. When a single image is processed with 2D convolution, the output is also an image. When 2D convolution operation is applied on multiple images, the output is a single image. When 3D convolution is applied on multiple images, the output feature map consists of more than one images. Therefore, 3D convolution preserves temporal information while 2D convolution loses it. 2D pooling operation downsamples a feature map in spatial dimensions. Therefore, the depth of the feature map does not change. 3D pooling operation downsamples a feature map both spatially and temporally which reduces the size of the depth of the feature map. 3D CNNs are proven to be very powerful feature extractors when pre-trained with a large video dataset. In other words, a pre-trained CNN can successfully be used on a new dataset without fine tuning. (Tran, Bourdev, Fergus, Torresani, & Paluri, 2015)

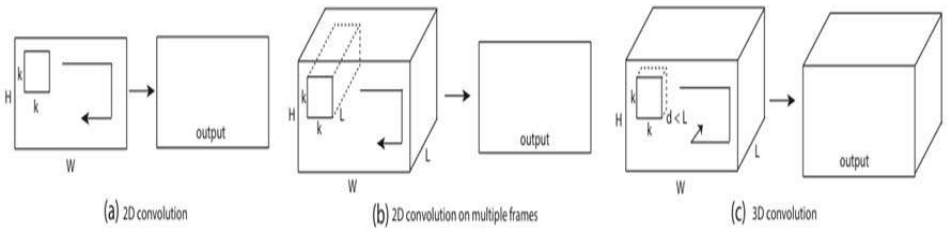

Figure 2.12. Illustration of 2D convolution and 3D convolution (Tran et al., 2015)

2.4.6. Two-Stream Inflated 3D Convolutional Neural Network (I3D)

3D CNNs are more suitable than 2D CNNs for video analysis because filters in their convolutional and pooling layers are spatio-temporal. Two-stream networks use not only RGB images but also optical flow frames which result in

more accuracy. Therefore, it is natural to expect that using 3D CNNs in a two-stream network will combine benefits of both 3D CNNs and two-stream networks. There are very big datasets and powerful architectures for 2D images. When 2D CNNs are used, it is straightforward to benefit from image datasets and architectures for video related tasks. For example, you can directly use an image classifier network for classifying frames. 2D CNNs can be inflated to 3D to benefit from image datasets and architectures. (Carreira & Zisserman, 2017)

The inflation operation is done as follows. Filters of convolutional layers are typically symmetric (N x N). The filters are kept symmetric in 3D (N x N x N). This is done by copying the 2D filters N times to make it 3D. Filters are rescaled by dividing with N. The transformation that is explained makes sure that generated activations are same for both single image and a video that is generated by repeating the same image. It is very common to use the same stride for spatial width and spatial height. When determining temporal stride for convolutional and pooling layers, choosing the same stride with spatial dimensions may not be a good choice. (Carreira & Zisserman, 2017)

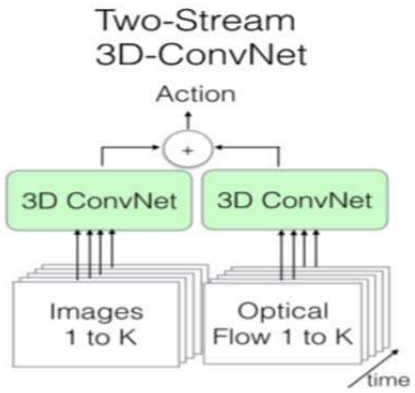

Figure 2.13. An illustration of a two-stream inflated 3D CNN. K represents number of frames in a video. (Carreira & Zisserman, 2017)

2.4.7. Deep Temporal Linear Encoding Networks

The deep learning based action recognition algorithms that are described above are not able to encode features from entire videos. They only operate on a single frame or a set of frames. TLE (Temporal Linear Encoding) is a video representation that encodes appearance and motion of entire video into features. This video representation is implemented by adding a new layer to convolutional neural networks. They can be integrated into different type of networks such as 2D CNNs, 3D CNNs and two-stream networks. TLE is illustrated in the figure below.

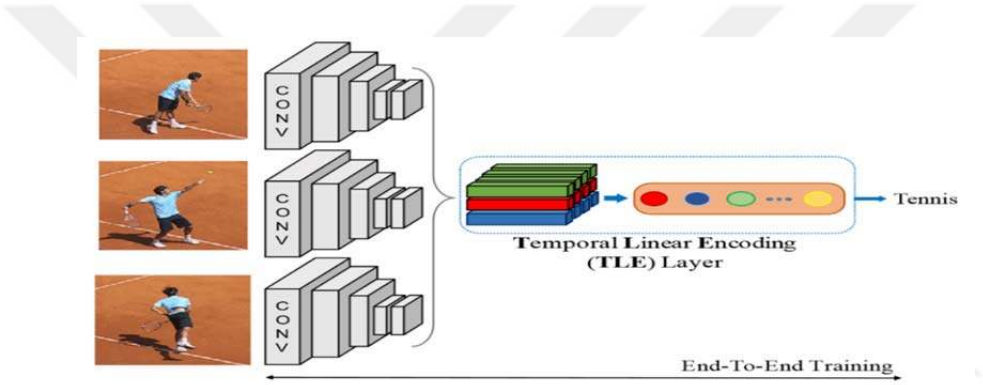

Figure 2.14. An illustration of temporal linear encoding video representation (Diba, Sharma, & Gool, 2017)

Temporal linear encoding layer receives the output feature maps of several neural networks. These neural networks have the same architecture and they share the same weights and biases. They are applied on different segments of video. These segments can be a single frame or a clip. The output feature map of the last convolutional layer is fed into temporal linear encoding layer. When TLE is used with two-stream neural networks, both spatial CNNs and temporal CNNs share trainable parameters. The scores of spatial stream and temporal stream are combined. Convolutional neural networks with TLE layer are trained end-to-end.

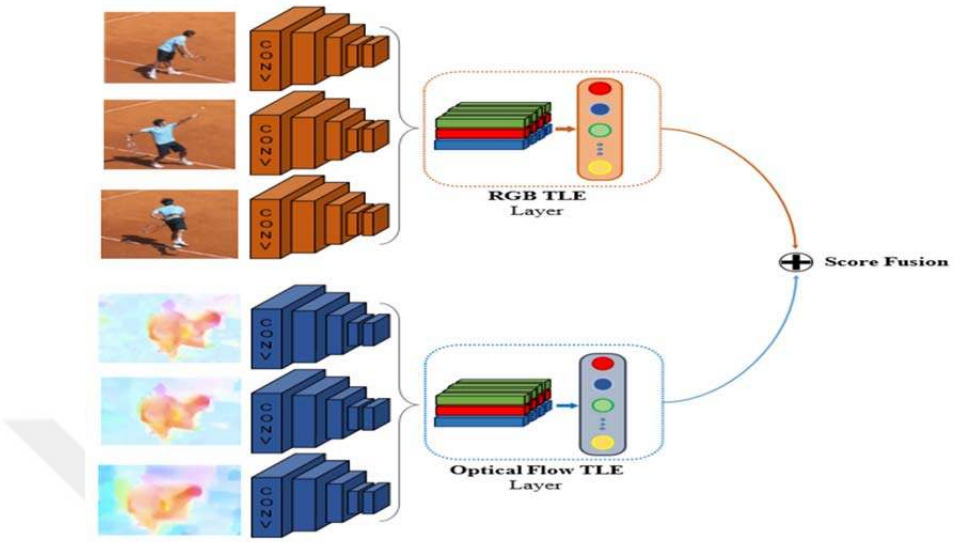

Figure 2.15. An illustration of a two-stream neural network with a separate TLE for spatial stream and temporal stream (Diba et al., 2017)

The TLE layer consists of two steps. The first step is aggregation of the feature maps that are received from different networks. The second step is encoding of the aggregated features. Aggregation of different feature maps can be done by element-wise average of feature maps, taking element-wise maximum of feature maps and element-wise multiplication of feature maps. Element-wise multiplication has been proven to be the best performing aggregation technique among these techniques. By aggregating different feature maps, we obtain a single feature map that represents visual and motion characteristics of entire video. The output of aggregation function is fed into an encoding function. The main advantage of encoding is that every channel of the input interacts with every other channel. Therefore, the output of the encoding function is a very powerful representation of the entire video.

There are two methods to encode the feature map. The first method is bilinear pooling. Consider that the input of the encoding algorithm is size of $h \times w \times c$ where h is the height, w is the width and c is the depth. Consider that this matrix

is reshaped to the matrix with size of *(hw) x c*. If the reshaped matrix is named as *X*, the bilinear pooling is computed as follows.

$$y = W[XX^T] \tag{2.12}$$

In the equation above, *y* is the output of the encoding method of size c^2. [] operation converts the matrix to a vector by concatenating columns. *W* represents the trainable parameters for encoding. Signed square root and L_2-normalization is applied on y.

The second method for encoding is using fully-connected pooling. A fully-connected layer can be placed between TLE and classification layer. Parameters of the fully-connected layer can be learned by end-to-end training. The last layer (classification layer) is the softmax layer in both encoding methods. These two methods were presented in the original paper of deep temporal linear encoding networks. However, different encoding methods in the literature such as deep fisher encoding (Tang et al., 2016) and VLAD (Arandjelovic, Gronat, Torii, Pajdla, Sivic, 2016) can also be used for encoding. (Diba et al., 2017)

2.4.8. Appearance-and-Relation Networks

A powerful video representation has to explore both visual and motion cues in the video. Different algorithms have different approaches for representing these cues. Two-stream networks use two different streams. Spatial stream explores visual cues and temporal stream explores motion cues. When the output scores of the two streams are combined, accuracy is expected to improve because two streams are complementary. However, calculating optical flow and training two different networks are very expensive in terms of time and computational power. 3D CNNs aim to detect these cues from volume of input frames implicitly. When 3D CNNs used in two-stream network, the accuracy improves significantly which indicates that 3D CNNs cannot benefit from all motion cues. Appearance-and-

Relation networks (ARTNet) have a unique way to detect these cues. An ARTNet consists of SMART (Simultaneously model appearance and relation) units. A SMART block consists of two different branches. The first branch is appearance branch which is responsible for detecting visual cues. The second branch is relation branch which is responsible for detecting motion cues. The output of two branches are later combined with concatenation and reduction. Therefore, SMART blocks are able to detect visual and motion cues explicitly and simultaneously. When multiple SMART blocks sequentially used in an ARTNet, early layers detect low level features and later layers detect high level features.

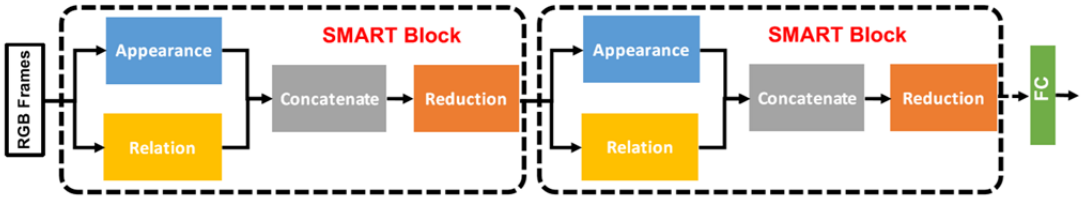

Figure 2.16. An illustration of an ARTNet (Wang, Li, Li, & Gool, 2018)

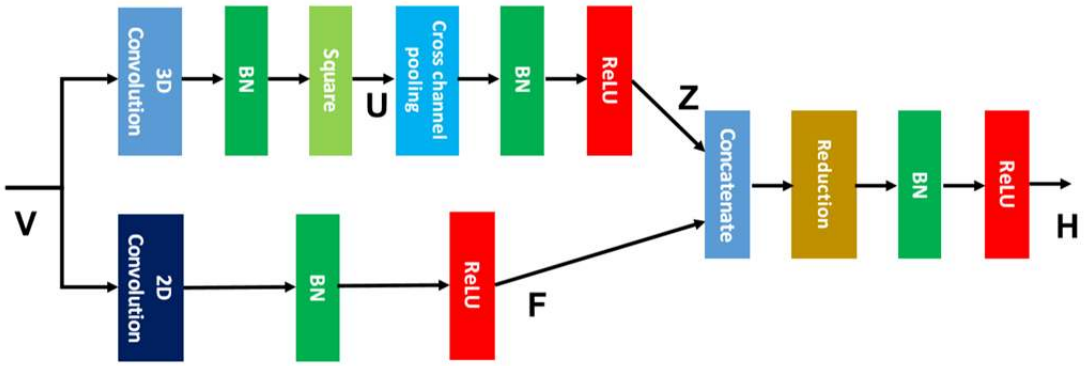

Figure 2.17. An illustration of a SMART unit. The output of appearance branch is denoted with F and the output of relation branch is denoted with Z. (Wang et al., 2018)

ARTNets receive a volume of frames as input similar to 3D CNNs. Appearance branch is necessary because objects existing in the frame and scene

characteristics can be critical for distinguishing different classes of actions. In appearance branch, the input is processed with a 2D convolutional layer. Then, batch normalization (Ioffe & Szegedy, 2015) and ReLU (Agarap, 2018) are applied to the output of the 2D convolutional layer. The size of the output of the appearance branch is W_s X H_s X T_s X C_s where W_s denotes the spatial width, H_s denotes the spatial height, T_s denotes the temporal depth and C_s denotes the number of kernels. Relation branch is responsible for capturing transformation and relation among frames. The relation and transformation are important because it contains motion cues. The first step of relation branch is applying 3D convolution on the received input volume. Then, batch normalization and square function are applied on the output of the 3D convolutional layer. After these steps, we obtain a matrix that is size of W_t X H_t X T_t X C_t. The next step is cross-channel pooling. The aim of cross-channel pooling is detecting relations among frames. It is implemented with a 1 x 1 x 1 convolutional layer. Cross-channel pooling outputs a matrix with the size of W_t X H_t X T_t X C_t'. The output of cross-channel pooling also goes through batch normalization and ReLU.

The output of appearance branch and relation branch are concatenated and a feature volume with the size of W' X H' X T' X (C_s + C_t') is obtained. Reduction operation is implemented with a 1 X 1 X 1 convolution. The output of reduction operation is size of W' X H' X T' X C_f. This reduced feature volume also goes through batch normalization and ReLU.

There are some rules that are defined for all ARTNets. The width, height and temporal depth of the output of two branches have to be the same to be able to concatenate the outputs. In other words, W_s = W_t = W', H_s = H_t = H', T_s = T_t= T'. The number of kernels in the 2D convolutional layer of the appearance branch has to be equal to the number of kernels in the 3D convolutional layer of the relation branch. Therefore, C_s = C_t. Cross-channel pooling is implemented with a convolution with the size of 1 x 1 x 1 x 2. In other words, C_t = 2C_t'. All the weights of the cross-channel pooling are fixed to be 0.5. The number of kernels in the

output of the SMART unit is equal to the number of kernels in the output of appearance branch. A 3D convolutional layer can easily be replaced by a SMART block by adjusting the parameters appropriately. (Wang et al., 2018)

2.4.9. Action Recognition using Visual Attention

Humans do not focus on the whole scene to observe it. Instead, they focus on the relevant parts of the scene. However, most of the action recognition algorithms fail to mimic this behavior of human visual system. Algorithms inspired from natural events and processes can be very successful. Therefore, attention models aim to focus on relevant areas of frames and videos. Attention mechanism has the advantage of being interpretable which is a property that deep neural networks usually do not have. By observing where the model focused on, one can express why the network correctly or incorrectly classified a sample. For example, in the images below, the network focused on the ball, the club and the player for recognizing golf swinging action and the network focused on the trampoline for recognizing trampoline jumping action.

Figure 2.18. Frames taken from two different videos in UCF-11 dataset (Liu, Luo, & Shah, 2009). White regions indicate the areas that the network focuses on. Brightness indicates the strength of the focus. (Sharma, Kiros, & Salakhutdinov, 2015)

The video frame is fed into a convolution neural network. The output feature map of the last convolutional layer is used in the model. Fully connected layers can be discarded. The feature map is shape of K X K X D. This array is

reshaped to be K^2 X D. In other words, we have K^2 different D dimensional vectors. This array is called as a feature cube. Each D dimensional vector is called as a feature slice.

$$X_t = [X_{t,1}, \ldots\ldots, X_{t,K^2}], \qquad X_{t,i} \in R^D$$

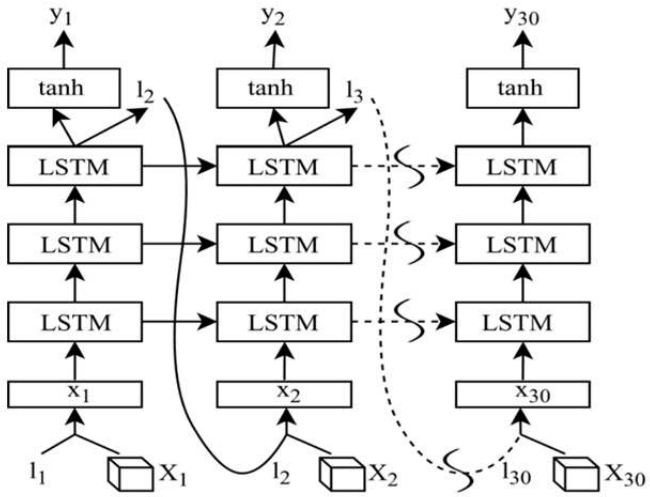

Figure 2.19. An illustration of visual attention based action recognition model (Sharma et al., 2015)

After convolutional layers, there are LSTM layers. The feature cube is not directly inputted into the next LSTM layer. The network focuses more on some feature slices and less on others. The strength of focus is determined by location softmax which uses the hidden states of the last LSTM layer in the previous time step. The location softmax is defined in equation 2.13.

$$l_{t,i} = \frac{exp(W_i^T h_{t-1})}{\sum_{j=1}^{K \times K} exp(W_j^T h_{t-1})} \qquad i \in 1\ldots..K^2 \qquad (2.13)$$

After l values which represent strength of focus is calculated, the attention mechanism computes the input for the first LSTM by the following equation.

$$x_t = \sum_{i=1}^{K^2} l_{t,i} X_{t,i} \tag{2.14}$$

The following equations (Ba et al., 2015) are used to initialize c_0 and h_0. The $f_{init,c}$ and $f_{init,h}$ are multilayer perceptrons and T is the number of time steps in the model.

$$c_0 = f_{init,c}\left(\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{K^2}\sum_{i=1}^{K^2} X_{t,i}\right)\right) \tag{2.15}$$

$$h_0 = f_{init,h}\left(\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{K^2}\sum_{i=1}^{K^2} X_{t,i}\right)\right) \tag{2.16}$$

The loss function of the model is defined below where y_t is one hot label vector, $\hat{y}_t$ is the vector of class probabilities at time step t, T is the total number of time steps, C is the number of output classes, λ is the attention penalty coefficient, γ is the weight decay coefficient, and θ represents all the model parameters.

$$L = -\sum_{t=1}^{T}\sum_{i=1}^{C} y_{t,i} \log \widehat{y_{t,i}} + \lambda \sum_{i=1}^{K^2}(1 - \sum_{t=1}^{T} l_{t,i})^2 + \gamma \sum_{i}\sum_{j} \theta_{i,j}^2 \tag{2.17}$$

The behavior of the attention mechanism depends on the attention penalty coefficient. In the Figure 2.20, you can observe that on sample frames taken from the same video. In the first row, the attention penalty coefficient is set to 0. In these frames, the model tends to focus on a fixed location. In the second row, the attention penalty coefficient is set to 1 and the model explores different areas. In the third row, the attention penalty coefficient is set to 10 and the model focuses on everywhere.

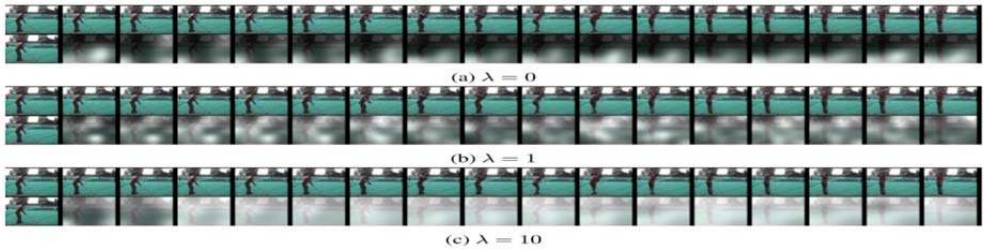

Figure 2.20. An illustration of effect of different attention penalty coefficients. (Sharma et al., 2015)

3. MATERIAL AND METHOD

3.1. Material

3.1.1. Programming Language and Framework

I will use Python programming language for development. Python offers several advantages for developing deep learning software. Firstly, Python is compatible with more deep learning frameworks than any other programming language. Secondly, it enables rapid development. I chose Tensorflow as my framework because of its performance and community support. (Abadi et al., 2016)

3.1.2. Libraries

3.1.2.1. NumPy

NumPy is a Python library that provides functions for creating and manipulating multi-dimensional arrays. Video data is kept in multi-dimensional arrays. Therefore, NumPy is a useful tool for the thesis. (Walt, Colbert, & Varoquaux, 2011)

3.1.2.2. Matplotlib

Matplotlib is a Python library for plotting different types of figures like histograms, bar charts and scatterplots. Plots are useful for both presenting data in dissertation and debugging the code. (Hunter, 2007)

3.1.2.3. OpenCV

OpenCV is a very popular open source computer vision library. It provides more than 2500 algorithms. The library has been used for two different tasks in the thesis. Firstly, videos were read by the functions of OpenCV. Secondly, optical flows were calculated with the functions of the OpenCV. (Culjak, Abram, Pribanic, Dzapo, & Cifrek, 2012)

3.1.3. UCF101 Dataset

Figure 3.1. Sample images from the UCF101 dataset (Soomro, Zamir, & Shah, 2012)

UCF101 dataset is a human action recognition dataset consisting of 101 action classes and 13320 videos. This dataset has two major advantages over other existing datasets. Firstly, most datasets include very low number of classes. Secondly, most of the actions in these datasets are carried out in strictly controlled circumstances. However, UCF101 dataset includes videos recorded in different lighting conditions and quality. Some example frames from the dataset can be seen in the figure 3.1. UCF101 dataset includes 101 classes. 50 classes were taken from UCF50 dataset. 51 classes are new (Soomro, Zamir, & Shah, 2012).

1. Classes from UCF50: Baseball Pitch, Basketball Shooting, Bench Press, Biking, Billiards Shot, Breaststroke, Clean and Jerk, Diving, Drumming, Fencing, Golf Swing, High Jump, Horse Race, Horse Riding, Hula Hoop, Javelin Throw, Juggling Balls, Jumping Jack, Jump Rope, Kayaking,

Lunges, Military Parade, Mixing Batter, Nunchucks, Pizza Tossing, Playing Guitar, Playing Piano, Playing Tabla, Playing Violin, Pole Vault, Pommel Horse, Pull Ups, Punch, Push Ups, Rock Climbing Indoor, Rope Climbing, Rowing, Salsa Spins, Skate Boarding, Skiing, Skijet, Soccer Juggling, Swing, Tai Chi, Tennis Swing, Throw Discus, Trampoline Jumping, Volleyball Spiking, Walking with a dog, Yo Yo

2. Newly added classes: Apply Eye Makeup, Apply Lipstick, Archery, Baby Crawling, Balance Beam, Band Marching, Basketball Dunk, Blow Drying Hair, Blowing Candles, Body Weight Squats, Bowling, Boxing-Punching Bag, Boxing-Speed Bag, Brushing Teeth, Cliff Diving, Cricket Bowling, Cricket Shot, Cutting In Kitchen, Field Hockey Penalty, Floor Gymnastics, Frisbee Catch, Front Crawl, Haircut, Hammering, Hammer Throw, Handstand Pushups, Handstand Walking, Head Massage, Ice Dancing, Knitting, Long Jump, Mopping Floor, Parallel Bars, Playing Cello, Playing Daf, Playing Dhol, Playing Flute, Playing Sitar, Rafting, Shaving Beard, Shotput, Sky Diving, Soccer Penalty, Still Rings, Sumo Wrestling, Surfing, Table Tennis Shot, Typing, Uneven Bars, Wall Pushups, Writing On Board

Table 3.1. Number of samples for each class in the UCF101 dataset

Class Name	**Training Samples**	**Testing Samples**	**Validation Samples**
ApplyEyeMakeup	102	29	14
ApplyLipstick	81	22	11
Archery	102	29	14
BabyCrawling	93	26	13
BalanceBeam	77	21	10
BandMarching	109	31	15
BaseballPitch	105	30	15
Basketball	94	26	13
BasketballDunk	92	26	13
BenchPress	112	32	16
Biking	95	26	13
Billiards	105	30	15
BlowDryHair	92	26	13
BlowingCandles	78	21	10
BodyWeightSquats	79	22	11
Bowling	109	31	15
BoxingPunchingBag	115	32	16
BoxingSpeedBag	95	26	13
BreastStroke	71	20	10
BrushingTeeth	92	26	13
CleanAndJerk	79	22	11
CliffDiving	98	27	13
CricketBowling	99	27	13
CricketShot	118	33	16
“CuttingInKitchen	77	22	11

Table 3.1.(Continue)

Diving	105	30	15
Drumming	113	32	16
Fencing	78	22	11
FieldHockeyPenalty	89	25	12
FloorGymnastics	88	25	12
FrisbeeCatch	89	25	12
FrontCrawl	97	27	13
GolfSwing	99	27	13
Haircut	91	26	13
Hammering	98	28	14
HammerThrow	105	30	15
HandstandPushups	91	25	12
HandstandWalking	78	22	11
HeadMassage	104	29	14
HighJump	87	24	12
HorseRace	88	24	12
HorseRiding	115	32	16
HulaHoop	88	25	12
IceDancing	112	31	15
JavelinThrow	83	23	11
JugglingBalls	85	24	12
JumpingJack	87	24	12
JumpRope	102	28	14
Kayaking	99	28	14
Knitting	87	24	12
LongJump	92	26	13
Lunges	90	25	12

Table 3.1.(Continue)

MilitaryParade	88	25	12
Mixing	96	27	13
MoppingFloor	77	22	11
Nunchucks	93	26	13
ParallelBars	81	22	11
PizzaTossing	80	22	11
PlayingCello	116	32	16
PlayingDaf	106	30	15
PlayingDhol	116	32	16
PlayingFlute	109	31	15
PlayingGuitar	112	32	16
PlayingPiano	74	21	10
PlayingSitar	111	31	15
PlayingTabla	78	22	11
PlayingViolin	70	20	10
PoleVault	106	29	14
PommelHorse	87	24	12
PullUps	70	20	10
Punch	112	32	16
PushUps	72	20	10
Rafting	78	22	11
RockClimbingIndoor	102	28	14
RopeClimbing	85	23	11
Rowing	97	27	13
SalsaSpin	94	26	13
ShavingBeard	113	32	16
Shotput	102	28	14

Table 3.1.(Continue)

SkateBoarding	84	24	12
Skiing	95	27	13
Skijet	70	20	10
SkyDiving	77	22	11
SoccerJuggling	104	29	14
SoccerPenalty	97	27	13
StillRings	79	22	11
SumoWrestling	82	23	11
Surfing	89	25	12
Swing	92	26	13
TableTennisShot	98	28	14
TaiChi	70	20	10
TennisSwing	117	33	16
ThrowDiscus	91	26	13
TrampolineJumping	85	23	11
Typing	96	27	13
UnevenBars	74	20	10
VolleyballSpiking	82	23	11
WalkingWithDog	87	24	12
WallPushups	91	26	13
WritingOnBoard	107	30	15
YoYo	91	25	12

3.1.4. HMDB Dataset

HMDB (The Human Motion Database) is a human action recognition dataset that tries to reflect the diversity of human actions. The clips in the dataset were collected by a group of students from various sources such as digitized

movies, Prelinger archive, YouTube videos and Google videos. They comply some restrictions. A clip has to include only one action. The height of main actor must be at least 60 pixels. A clip cannot be shorter than 1 second. A minimum contrast level was set.

The initial version of dataset included 60 classes. The classes having less than 101 clips were removed from dataset. The current dataset includes 51 classes and 6766 videos. See figure 3.2 for sample images. The action classes can be divided into 5 groups (Kuehne, Jhuang, Garrote, Poggio, & Serre, 2011).

Figure 3.2. Sample images from the HMDB dataset. From top-left to bottom-right, actions are: hand-waving, drinking, sword fighting, diving, running and kicking. (Kuehne, Jhuang, Garrote, Poggio, & Serre, 2011)

1. General facial action

   Smile, laugh, chew, talk

2. Facial actions with object manipulation

   Smoke, eat, drink

3. General body movements

   Cartwheel, clap hands, climb, climb stairs, dive, fall on the floor, backhand flip, handstand, jump, pull up, push up, run, sit down, sit up, somersault, stand up, turn, walk, wave

4. Body movements with object interaction

   Brush hair, catch, draw sword, dribble, golf, hit something, kick ball, pick, pour, push something, ride bike, ride horse, shoot ball, shoot bow, shoot gun, swing baseball bat, sword exercise, throw

5. Body movements with object interaction

   Fencing, hug, kick someone, kiss, punch, shake hands, sword fight

Table 3.2. Number of samples for each class in the HMDB dataset

Class Name	**Training Samples**	**Testing Samples**	**Validation Samples**
brush_hair	75	21	10
cartwheel	76	21	10
catch	72	20	10
chew	78	21	10
clap	91	26	13
climb	77	21	10
climb_stairs	79	22	11
dive	89	25	12
draw_sword	73	20	10
dribble	102	29	14
drink	116	32	16
eat	77	21	10
fall_floor	96	27	13
fencing	82	23	11
flic_flac	76	21	10
golf	74	21	10
handstand	80	22	11
hit	90	25	12
hug	84	23	11
jump	106	30	15
kick	91	26	13
kick_ball	91	25	12
kiss	71	20	10
laugh	91	25	12
pick	75	21	10

Table 3.2.(Continue)

pour	75	21	10
pullup	74	20	10
punch	89	25	12
push	82	23	11
pushup	73	20	10
ride_bike	73	20	10
ride_horse	82	23	11
run	163	46	23
shake_hands	114	32	16
shoot_ball	92	26	13
shoot_bow	79	22	11
shoot_gun	73	20	10
sit	100	28	14
situp	74	21	10
smile	72	20	10
smoke	78	21	10
somersault	98	28	14
stand	109	30	15
swing_baseball	101	28	14
sword	90	25	12
sword_exercise	90	25	12
talk	84	24	12
throw	72	20	10
turn	168	48	24
walk	385	109	54
wave	74	20	10

3.1.5. Hollywood2 Dataset

Figure 3.3. Some example images from the dataset (Laptev, Marszalek, Schmid & Rozenfeld, 2008)

Hollywood2 dataset is a human action recognition dataset that consists of video clips extracted from Hollywood films. Firstly, a text classifier was trained on movie scripts for automatic labeling of action classes. Then, video clips for action classes were extracted using 69 movie scripts. For training and testing, different movies were used. After extracting videos, the testing set was manually checked to validate and correct classes assigned to videos. The training set remained as it is. In other words, wrong labels were not corrected in the training set. The total length of the videos is 600k frames and 7 hours of video. The dataset also contains scene classes for using correlation between scene classes and action classes. The distribution of videos according to classes is given in table 3.3 (Marszalek, Laptev, & Schmid, 2009). Some example frames from the dataset is illustrated in figure 3.3.

Table 3.3. Number of samples for each class in Hollywood2 dataset

Class Name	Training Samples	Testing Samples	Validation Samples
AnswerPhone	83	23	11
DriveCar	126	36	18
Eat	46	12	6
FightPerson	72	20	10
GetOutCar	71	20	10
HandShake	46	12	6
HugPerson	48	13	6
Kiss	146	41	20
Run	189	53	26
SitDown	142	40	20
SitUp	41	11	5
StandUp	196	55	27

3.2. Method

3.2.1 Proposed Method

For my thesis, I propose modified recurrent convolutional neural networks for detecting actions in videos. A video consists of frames which are images that form the video. Videos are very suitable to be processed by modified recurrent convolutional neural networks. Because, each frame has semantic dependence on previous frames. For example, a video of a penalty kick consists of a few different parts. In the beginning, a player will prepare and kick the ball. After that, the ball will move and the goalkeeper will make a move to catch the ball. It may not be possible to understand the action without considering the first part of the video.

The main aim of the proposed methods is to use information from previous frames to classify a frame. To accomplish this task, modified recurrent convolutional neural networks are utilized. Two different approaches are proposed

to combine information between frames. These approaches are explained in more details in the following sections.

3.2.1.1 Modified Recurrent Convolutional Neural Networks for Action Recognition

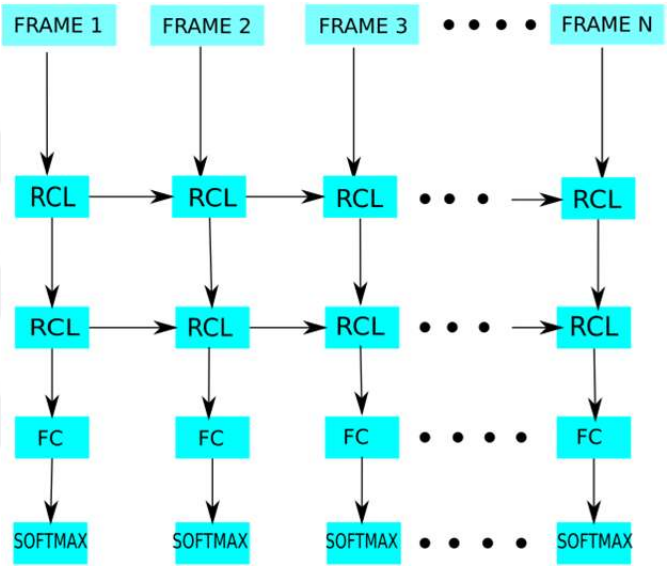

Figure 3.4. Modified RCNN for action recognition

In the proposed modified recurrent convolutional neural network, each recurrent convolutional layer (RCL) receives the output of the same layer for the previous frame and the output of the previous layer of the same frame. The proposed network is inspired from the recurrent convolutional neural network presented in section 2.3. As described in section 2.3, recurrent convolution layers receive two different inputs. Those are feed-forward input and recurrent input. The output of the previous layer is the feed-forward input. This is the only input in the standard convolutional layer. The output of the same layer for the previous frame is the recurrent input. Convolution operation is applied for recurrent input and feed-forward input with separate weights and biases. The result of convolution on feed-

forward input is multiplied with 0.7 and the result of convolution on recurrent input is multiplied with 0.3. After that, the result of two multiplications are added to find the output of the recurrent convolutional layer. The output of the RCL has to carry the information of both the previous layer and the current layer. The first convolution is multiplied by 0.7 and the second one is multiplied by 0.3 because information from previous frames should not completely replace information from current frame. In initial experiments, convolution operations were not multiplied with any value and the accuracy was lower. The best values used for multiplication depend on the network and the data. In future work, more experiments can be done on these values to have more information about how to select these parameters. Same weight and biases are used for all frames.

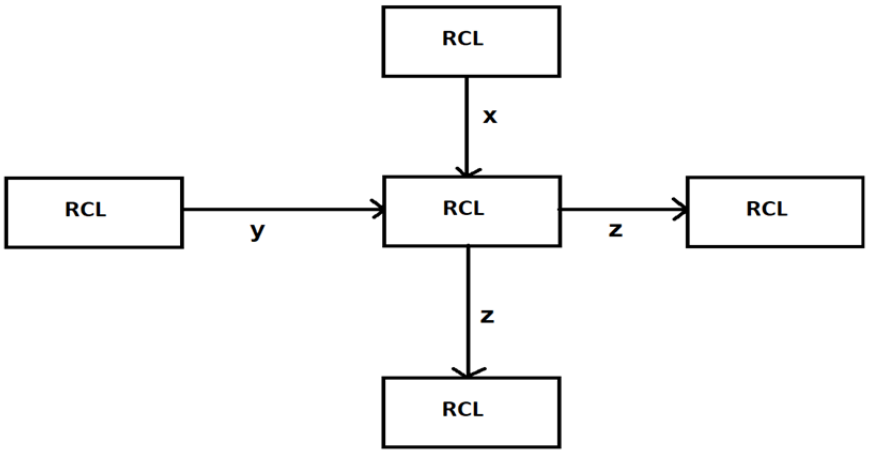

Figure 3.5. A figure illustrating how an RCL operates

In the figure 3.5, x represents the feed-forward input, y represents the recurrent input and z represents the output of the recurrent convolutional layer (RCL) in the middle. The value of z can be calculated using the equation below.

$$z = \text{conv}(x, w_f n_L, b_f n_L) * 0.7 + \text{conv}(y, w_r n_L, b_r n_L) * 0.3 \quad (3.1)$$

The w_fn_L is the weight for feed-forward input for the Lth layer. The b_fn_L is the bias for the feed-forward input for the Lth layer. The w_rn_L is the weight for the recurrent input for the Lth layer. The b_rn_L is the bias for the recurrent input for the Lth layer. Those weights and biases are shared for different frames. In other words, the weights and biases for a specific layer are same for all frames. Each layer includes two convolution operation. It is expected that adding convolution operation on recurrent input will add beneficial information from previous frames.

The result of two convolutions are summed. Therefore, the size of the results must be the same to be able to sum them. For the first frame, there are not any recurrent inputs available. Therefore, a matrix consisting of zeros is used to be able to make the calculations. This type of recurrence is investigated for only convolutional layers in this study. For future work, fully-connected layers and pooling layers can also be investigated.

3.2.1.2. Feature Concatenating CNN

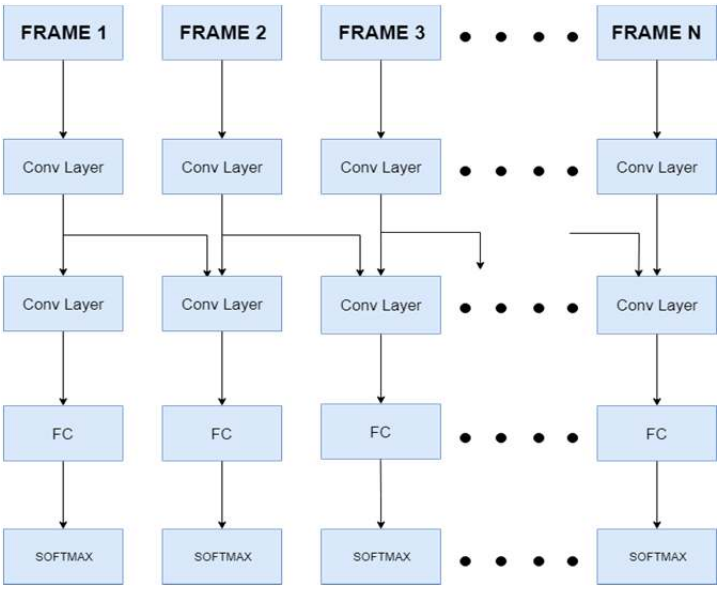

Figure 3.6. A figure illustrating feature concatenating CNN

The main aim of feature concatenating CNN is same with modified recurrent convolutional neural networks. They both incorporate information from previous frames. However, feature concatenating CNN is not as sophisticated as modified recurrent CNN. In feature concatenating CNN, each layer has only 1 convolution operation instead of 2. All convolutional layers except the first layer receive two inputs. Those are the output of previous layer and the output of previous layer for the previous frame. Those outputs are concatenated. If the feature map of the previous convolutional layer has 10 filters, the input of the current convolutional layer will receive 20 filters because 10 filters will be coming from the previous frame. Convolution operation is applied on concatenated input. In the first frame, layers cannot receive input from the previous frame because there are not any previous frames. Therefore, they are fed with zeros. In this study, feature concatenation has only been explored for convolutional layers. However, fully-connected layers and pooling layers can be considered for future work.

3.2.2. Selected Parameters and Settings

For the training, Adam optimization algorithm has been used (Kingma & Ba, 2015). The default parameters of Tensorflow have not been altered for Adam algorithm. The initial learning rate has been set to 10^{-5}. It has been divided by 10 in every 8 epochs. ReLU activation function has been used after any convolution operation in all networks (Agarap, 2018). In modified recurrent CNN, each layer includes 2 convolution operation. The outputs of convolution operations are added after ReLU activation is applied on each convolution operation. L2 regularization (Laarhoven, 2017) is used in all networks. Beta value is selected to be 0.002. Frames are resized to 100x100x3 in all networks and in all datasets.

During training, 20 frames have been used from each video. Therefore, 20 frames have been sampled from each video. There is a constant number of frames between two sampled frames and it is called as step size. Step size is calculating by dividing the total number of frames by 20. For example, if a video has 40 frames,

the step size will be 2 because 20 frames will be sampled from 40 frames. The first frame of the video will be the first sampled frame. The third frame of the video will be the second sampled frame. The fifth frame of the video will be the third sampled frame. Videos having less than 20 frames are discarded.

Only one frame is inputted to the networks at a time. Frames that are sampled from the same video are inputted to the networks consecutively. When a frame is inputted to the network, the gradients are calculated. Gradients for all frames that belongs to the same video is summed. Then, the training algorithm is run using the sum of the gradients. This training strategy is used because it is suitable for modified recurrent CNN and feature concatenating CNN. With this training strategy, feature maps can be transmitted between sampled frames easily which is required by modified recurrent CNN and feature concatenating CNN. A different training strategy has also been investigated. Each time a number of frames with the same order have been sampled from different videos. For example, if batch size is 20, the first sampled frames from 20 different videos form the batch for the first epoch. For the second epoch, the second sampled frames are used. This approach did not give satisfying results in the experiments. Because, in each epoch, the network learns to classify frames at a specific position. When training is complete, the network can only classify frames that are located at the end of the video.

Some of the neural networks of this study receive optical flow as input. For optical flow, calcOpticalFlowFarneback function of OpenCV has been used. This function uses Gunnar Farneback (Farnebäck, 2003) algorithm for calculating optical flow. The function receives two frames and returns the optical flow between two frames. An optical flow matrix is 2-dimensional. Consider that the sampled frame is named as A and the following frames are named B, C and D respectively. The optical flow input for the sampled frame is calculated as follows. The first optical flow calculation is done between A and B. The second optical flow calculation is done between B and C. The third optical flow calculation is

done between C and D. The results of these three calculations are concatenated to form the input for the neural network. Therefore, the optical flow input is 6-dimensional. These neural networks receive this 6-dimensional input instead of 3-dimensional sampled frame. If the sampled frame is not followed by enough number of frames to calculate the optical flow because the sampled frame is one of the last frames of the video, the closest preceding frame that can be used for calculating optical flow is sampled instead of initially selected frame.

The 3D neural networks receive a number of consecutive frames instead of a single frame. Normally, the sampled frame is fed into the network as input. However, for 3D neural networks, the sampled frame and the 7 consecutive frames are concatenated to form the input. If the sampled frame is not followed by enough number of frames to form the input because the sampled frame is one of the last frames of the video, the closest preceding frame that is followed by at least 7 frames is sampled instead of initially selected frame.

3.2.3. Network Architectures

In this section, the details of the networks that are used in the tests have been shared. The number of filters in the layers of modified recurrent and feature concatenating networks has been selected to be less than that of standard networks because the number of trainable parameters in different networks are kept similar. In all networks, a softmax classification layer is placed after the last fully connected layer.

Table 3.4. Network for single-frame CNN

Layer Name	**Parameters**
Conv Layer #1	5x5, 96 filters, strides are 1
Pooling Layer #1	2x2, strides are 2
Conv Layer #2	3x3, 120 filters, strides are 1
Pooling Layer #2	2x2, strides are 2
Conv Layer #3	3x3, 180 filters, strides are 1
Pooling Layer #3	2x2, strides are 2
Conv Layer #4	3x3, 240 filters, strides are 1
Pooling Layer #4	2x2, strides are 2
Conv Layer #5	3x3, 256 filters, strides are 1
Pooling Layer #5	2x2, strides are 2
Fully Connected Layer #1	1024 neurons
Fully Connected Layer #2	512 neurons
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset

Table 3.5. Network for modified recurrent CNN

Layer Name	**Parameters**	**Recurrent Layer Parameters**
Conv Layer #1	5x5, 32 filters, strides are 1	5x5, 32 filters, strides are 1
Pooling Layer #1	2x2, strides are 2	
Conv Layer #2	3x3, 64 filters, strides are 1	3x3, 64 filters, strides are 1
Pooling Layer #2	2x2, strides are 2	
Conv Layer #3	3x3, 96 filters, strides are 1	3x3, 96 filters, strides are 1
Pooling Layer #3	2x2, strides are 2	
Conv Layer #4	3x3, 128 filters, strides are 1	3x3, 128 filters, strides are 1
Pooling Layer #4	2x2, strides are 2	
Conv Layer #5	3x3, 256 filters, strides are 1	3x3, 256 filters, strides are 1
Pooling Layer #5	2x2, strides are 2	
Fully Connected Layer #1	1024 neurons	
Fully Connected Layer #2	512 neurons	
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset	

Table 3.6. Network for feature concatenating CNN

Layer Name	**Parameters**
Conv Layer #1	5x5, 16 filters, strides are 1
Pooling Layer #1	2x2, strides are 2
Conv Layer #2	3x3, 40 filters, strides are 1
Pooling Layer #2	2x2, strides are 2
Conv Layer #3	3x3, 92 filters, strides are 1
Pooling Layer #3	2x2, strides are 2
Conv Layer #4	3x3, 200 filters, strides are 1
Pooling Layer #4	2x2, strides are 2
Conv Layer #5	3x3, 256 filters, strides are 1
Pooling Layer #5	2x2, strides are 2
Fully Connected Layer #1	1024 neurons
Fully Connected Layer #2	512 neurons
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset

Table 3.7. Network for single frame CNN (optical flow input)

Layer Name	**Parameters**
Conv Layer #1	5x5, 64 filters, strides are 1
Pooling Layer #1	2x2, strides are 2
Conv Layer #2	3x3, 128 filters, strides are 1
Pooling Layer #2	2x2, strides are 2
Conv Layer #3	3x3, 180 filters, strides are 1
Pooling Layer #3	2x2, strides are 2
Conv Layer #4	3x3, 220 filters, strides are 1
Pooling Layer #4	2x2, strides are 2
Conv Layer #5	3x3, 256 filters, strides are 1
Pooling Layer #5	2x2, strides are 2
Fully Connected Layer #1	1024 neurons
Fully Connected Layer #2	512 neurons
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset

Table 3.8. Network for modified recurrent CNN (optical flow input)

Layer Name	**Parameters**	**Recurrent Layer Parameters**
Conv Layer #1	5x5, 16 filters, strides are 1	5x5, 16 filters, strides are 1
Pooling Layer #1	2x2, strides are 2	
Conv Layer #2	3x3, 32 filters, strides are 1	3x3, 32 filters, strides are 1
Pooling Layer #2	2x2, strides are 2	
Conv Layer #3	3x3, 80 filters, strides are 1	3x3, 80 filters, strides are 1
Pooling Layer #3	2x2, strides are 2	
Conv Layer #4	3x3, 180 filters, strides are 1	3x3, 180 filters, strides are 1
Pooling Layer #4	2x2, strides are 2	
Conv Layer #5	3x3, 256 filters, strides are 1	3x3, 256 filters, strides are 1
Pooling Layer #5	2x2, strides are 2	
Fully Connected Layer #1	1024 neurons	
Fully Connected Layer #2	512 neurons	
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset	

Table 3.9. Network for feature concatenating CNN (optical flow input)

Layer Name	**Parameters**
Conv Layer #1	5x5, 16 filters, strides are 1
Pooling Layer #1	2x2, strides are 2
Conv Layer #2	3x3, 40 filters, strides are 1
Pooling Layer #2	2x2, strides are 2
Conv Layer #3	3x3, 92 filters, strides are 1
Pooling Layer #3	2x2, strides are 2
Conv Layer #4	3x3, 200 filters, strides are 1
Pooling Layer #4	2x2, strides are 2
Conv Layer #5	3x3, 256 filters, strides are 1
Pooling Layer #5	2x2, strides are 2
Fully Connected Layer #1	1024 neurons
Fully Connected Layer #2	512 neurons
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset

Table 3.10. Network for 3D CNN

Layer Name	**Parameters**
Conv Layer #1	5x5x3, 32 filters, strides are 1
Pooling Layer #1	2x2x1, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Conv Layer #2	3x3x3, 64 filters, strides are 1
Pooling Layer #2	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Conv Layer #3	3x3x3, 96 filters, strides are 1
Pooling Layer #3	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Conv Layer #4	3x3x3, 180 filters, strides are 1
Pooling Layer #4	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Conv Layer #5	3x3x3, 256 filters, strides are 1
Pooling Layer #5	2x2x1, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Fully Connected Layer #1	1024 neurons
Fully Connected Layer #2	512 neurons
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset

Table 3.11. Network for modified recurrent 3D CNN

Layer Name	**Parameters**	**Recurrent Layer Parameters**
Conv Layer #1	5x5x3, 16 filters, strides are 1	5x5, 16 filters, strides are 1
Pooling Layer #1	2x2x1, strides are 2 for the spatial dimensions and 1 for the temporal dimension	
Conv Layer #2	3x3x3, 32 filters, strides are 1	3x3, 32 filters, strides are 1
Pooling Layer #2	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension	
Conv Layer #3	3x3x3, 64 filters, strides are 1	3x3, 64 filters, strides are 1
Pooling Layer #3	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension	
Conv Layer #4	3x3x3, 128 filters, strides are 1	3x3, 128 filters, strides are 1
Pooling Layer #4	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension	
Conv Layer #5	3x3x3, 256 filters, strides are 1	3x3, 256 filters, strides are 1
Pooling Layer #5	2x2x1, strides are 2 for the spatial dimensions and 1 for the temporal dimension	
Fully Connected Layer #1	1024 neurons	
Fully Connected Layer #2	512 neurons	
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset	

Table 3.12. Network for feature concatenating 3D CNN

Layer Name	**Parameters**
Conv Layer #1	5x5x3, 8 filters, strides are 1
Pooling Layer #1	2x2x1, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Conv Layer #2	3x3x3, 24 filters, strides are 1
Pooling Layer #2	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Conv Layer #3	3x3x3, 60 filters, strides are 1
Pooling Layer #3	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Conv Layer #4	3x3x3, 144 filters, strides are 1
Pooling Layer #4	2x2x2, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Conv Layer #5	3x3x3, 256 filters, strides are 1
Pooling Layer #5	2x2x1, strides are 2 for the spatial dimensions and 1 for the temporal dimension
Fully Connected Layer #1	1024 neurons
Fully Connected Layer #2	512 neurons
Fully Connected Layer #3	The number of neurons is equal to number of classes in the dataset

4. RESULTS AND DISCUSSIONS

4.1. Testing Settings

Frames are sampled from the videos using the same way that is explained in section 3.2.2. The 20 sampled frames are classified separately in all videos. Based on predictions of the frames, predictions for the videos are calculated. We report two different accuracies for each method. The first metric of accuracy is overall accuracy. Overall accuracy is calculated by taking the most common prediction among the sampled frames in a video as the prediction for a video. The second metric of accuracy is accuracy based on the last frame. It is calculated by taking the prediction of the last sampled frame in a video as the prediction for a video. Discussions on the results are held on overall accuracies because in action recognition studies, the most common metric for evaluation is video accuracy.

9 different neural networks were trained and tested for this work. The details of these networks have been shared in the previous section. The networks that receive optical flow input are renamed as optical flow CNN, optical flow modified recurrent CNN and optical flow feature concatenating CNN. Additionally, results for 3 different two-stream networks were reported. Those are named as two-stream CNN, two-stream feature concatenating CNN, two-stream modified recurrent CNN. The output of a two-stream network is calculated by averaging softmax scores of the two networks. The output of the two-stream CNN is calculated by averaging softmax scores of the single-frame CNN and the optical flow CNN. The output of the two-stream modified recurrent CNN is calculated by averaging softmax scores of the modified recurrent CNN and the optical flow modified recurrent CNN. The output of the two-stream feature concatenating network is calculated by averaging the feature concatenating CNN and the optical flow feature concatenating CNN. Additionally, baseline models were generated for all datasets by randomly generating predictions for all videos. The accuracy of the baseline model is the percentage of the videos that are predicted correctly.

Apart from 9 single neural networks and 3 two-stream neural networks, 2 additional results are also reported. Those are calculated by averaging softmax scores of the multiple neural networks. For combination of all networks, the softmax scores of 9 neural networks are averaged. For combination of all networks except 3D networks, the softmax scores of 6 neural networks (all neural networks except 3D networks) are averaged. Additionally, accuracy and loss for each epoch in the validation sets of the datasets are reported with graphs.

The methods proposed in this study may not always increase the accuracy. The performance gain depends on the dataset and the type of the network. In the following subsections, the performances are examined. Since, we have kept the number of trainable parameters similar, we can compare the networks. Our networks, are not as big as those in the most studies of the literature and complicated testing strategies were not used because of hardware limitations. Therefore, our results are not very competitive. For example, in the UCF-101 dataset, an accuracy of %41.3 has been obtained without fine-tuning in (Karpathy et al., 2014) while the maximum accuracy obtained is %32.585 in this thesis. In the state-of-the-art methods such as (Carreira & Zisserman, 2017), accuracies that are above %90 are obtained in the UCF-101 dataset.

4.2. Hollywood2 Dataset

Videos in the Hollywood2 dataset are relatively longer than the other datasets. This fact affects the performance of the networks. Modified recurrent networks learn to benefit from relation between frames. In longer videos, the step size is larger which is an advantage for the modified recurrent networks because larger step size means more difference between frames. For validation accuracy, the modified recurrent networks outperform other networks in the same category (networks that receive the same input form a category) except for the 3D networks. 3D feature concatenating network and 3D modified recurrent network have smaller number of filters because we tried to keep number of trainable parameters similar.

Although this disadvantage also exists in other feature concatenating and modified recurrent networks, it is more severe for the 3D networks.

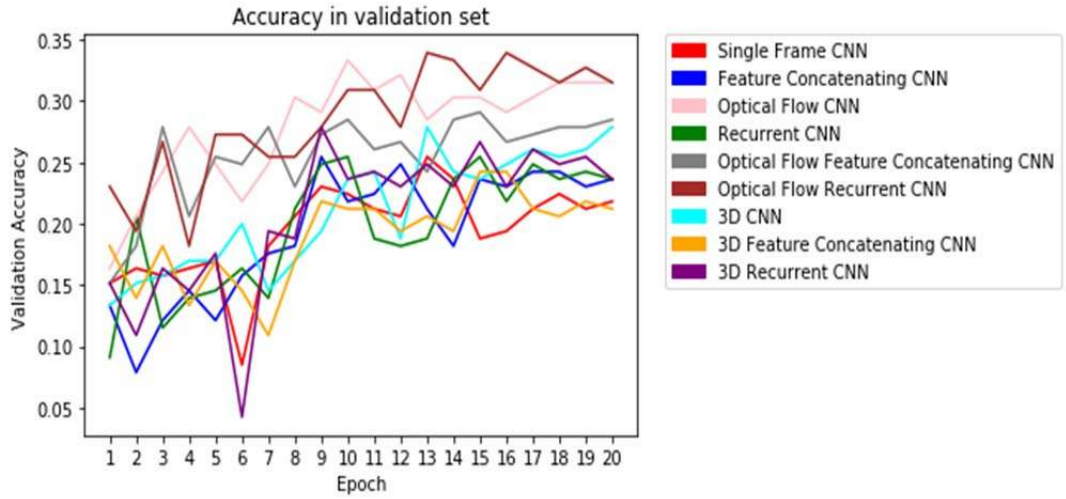

Figure 4.1. Validation Accuracies for the Hollywood2 dataset

The superior performance of the modified recurrent neural networks continue for the validation loss values. The loss values of all modified recurrent neural networks in all categories at the last epoch are less than the loss values of the other networks. The accuracies and loss values may not always have inverse proportion as expected. For example, the loss value of the 3D modified recurrent neural network at the last epoch is smaller than that of 3D CNN. On the other hand, 3D CNN has larger validation accuracy.

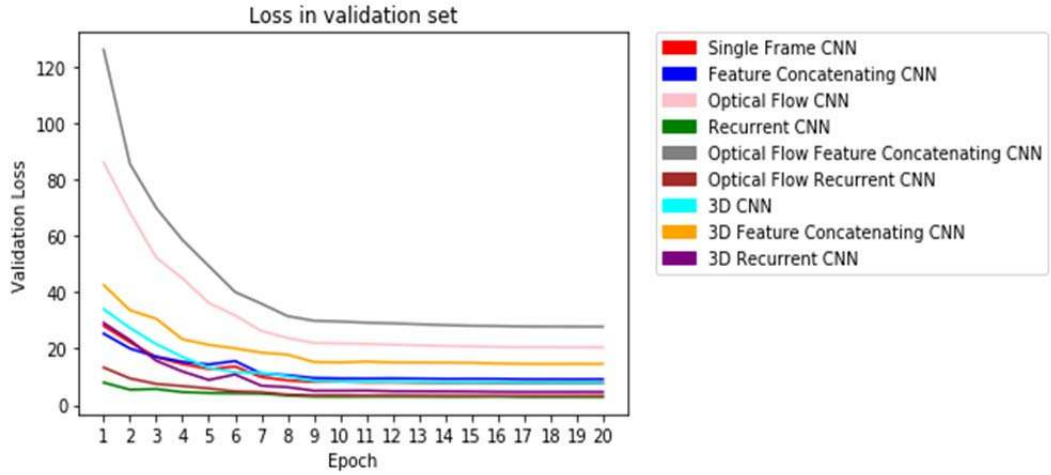

Figure 4.2. Validation Losses for the Hollywood2 dataset

For Hollywood2 dataset, our methods can provide better accuracy in all categories. In the category of single frame network, both of our methods provide better accuracy than others. In the optical flow networks category, optical flow modified recurrent CNN provides the best accuracy while optical flow feature concatenating network is behind optical flow CNN. Additionally, optical flow modified recurrent CNN is the best performing method in our tests. It also indicates that optical flow input is a very powerful representation for this dataset because the videos are longer and the optical flow displacement fields between frames carry more information when the videos are longer. Surprisingly, 3D feature concatenating network provides the best accuracy in the category of 3D networks. The input for the 3D networks is a few consecutive frames. Modified recurrent networks learn the relation between frames. However, in 3D networks, they need learn the relation between two array of frames which is harder. On the other hand, the feature concatenating neural network just concatenate feature maps from different frames and it is not more difficult to do in 3D networks. It is important to note that the way modified recurrent networks learn the relation between frames is more sophisticated than feature concatenating networks. Therefore, modified

recurrent networks outperform feature concatenating networks most of the time. For two-stream networks, two-stream modified recurrent network is the best performing one as expected. Combining multiple networks may not always provide the best performance. For Hollywood2 dataset, the combination of multiple networks does not outperform optical flow modified recurrent CNN.

Table 4.1. Accuracies in the test set of the Hollywood2 dataset.

Name of the Method	**Overall Accuracy**	**Accuracy Based on the Last Frame**
Single-Frame CNN	%16.964	%11.905
Feature Concatenating CNN	%18.452	%15.179
Modified Recurrent CNN	%22.024	%16.667
Optical Flow CNN	%25.893	%17.857
Optical Flow Feature Concatenating CNN	%24.107	%15.179
Optical Flow Modified Recurrent CNN	%31.25	%19.94
3D CNN	%22.024	%13.988
3D Feature Concatenating CNN	%23.512	%16.369
3D Modified Recurrent CNN	%18.75	%13.095
Two-Stream CNN	%24.702	%16.369
Two-Stream Feature Concatenating CNN	%26.19	%14.881
Two-Stream Modified Recurrent CNN	%30.357	%18.75
Combination of All Networks	%28.571	%20.833
Combination of All Networks Except 3D Networks	%29.464	%19.643
Baseline Result	%9.226	-

4.3. HMDB51 Dataset

HMDB51 dataset is a challenging dataset because of a few reasons. Firstly, there are some classes that are difficult to distinguish from each other. For example, smiling and laughing classes are very difficult to distinguish. Secondly, the main difference between different classes in the dataset is motion. In most of the datasets, using just the joint locations is enough for classification. However, it is not possible to obtain high accuracies without motion information in the HMDB51 dataset (Kuehne et al., 2011). This fact affected the performance of our tested methods. Our feature concatenating and modified recurrent networks could not outperform the standard networks except for the optical flow networks in the validation set. Optical flow modified recurrent CNN outperforms optical flow CNN and optical flow feature concatenating CNN. This indicates that our proposed networks cannot learn relation in terms of motion between frames unless optical flow is used as input. In future work, this can be worked on. Feature concatenating networks fail to outperform standard networks in all categories. This indicates that the layers of the feature concatenating networks could not learn the relations between frames.

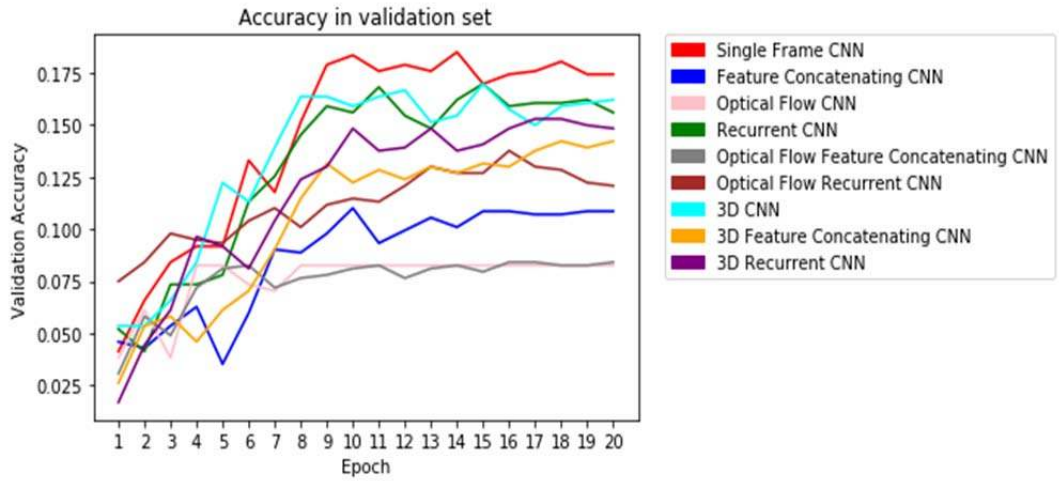

Figure 4.3. Validation Accuracies for the HMDB51 dataset.

The modified recurrent networks obtain the lowest losses while feature concatenating networks are behind other networks. These results are consistent with the accuracies on the test set.

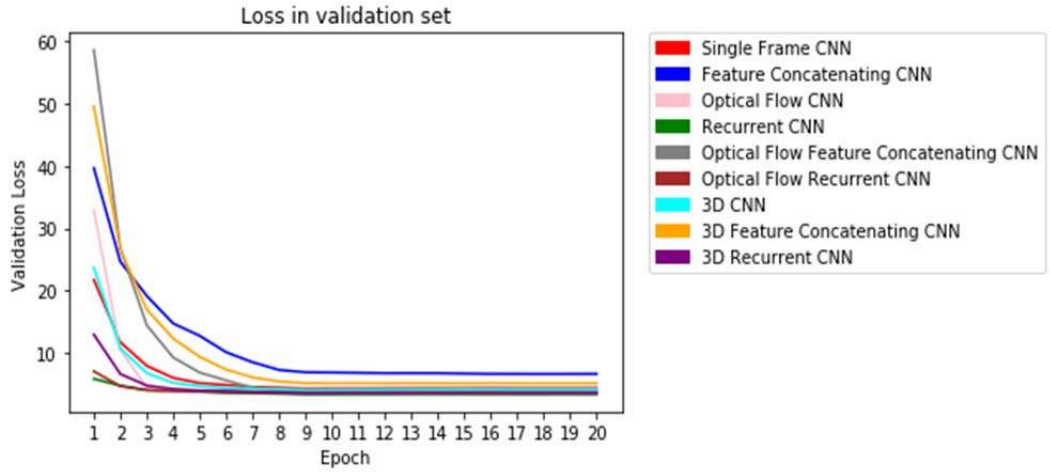

Figure 4.4. Validation Losses for the HMDB51 dataset.

Modified recurrent networks outperform others in all categories while feature concatenating networks is the worst performing in all categories in the test set. The results are different than validation results because of characteristics of different sets. The reason behind these results is that modified recurrent CNNs can learn motion relation between frames while feature concatenating CNNs cannot learn them in the test set. Feature concatenating CNN relies on feature concatenation for relation learning. Layers are expected to learn from concatenated feature maps while modified recurrent networks combine these feature maps in a more sophisticated way. In this dataset, the relation that need to be learned between frames is mostly motion instead of poses. Feature concatenating is not enough for this task. Two-stream modified recurrent CNN outperforms other two-stream networks as expected. The best result in this dataset is obtained by combining all networks.

Table 4.2. Accuracies in the test set of the HMDB51 dataset.

Name of the Method	**Overall Accuracy**	**Accuracy Based on the Last Frame**
Single-Frame CNN	%12.153	%08.927
Feature Concatenating CNN	%11.102	%08.327
Modified Recurrent CNN	%14.029	%11.178
Optical Flow CNN	%08.552	%08.252
Optical Flow Feature Concatenating CNN	%07.802	%07.202
Optical Flow Modified Recurrent CNN	%11.928	%10.278
3D CNN	%11.928	%09.752
3D Feature Concatenating CNN	%09.152	%06.827
3D Modified Recurrent CNN	%12.378	%10.128
Two-Stream CNN	%12.678	%08.927
Two-Stream Feature Concatenating CNN	%11.253	%08.402
Two-Stream Modified Recurrent CNN	%17.179	%14.029
Combination of All Networks	%17.704	%14.404
Combination of All Networks Except 3D Networks	%16.279	%11.853
Baseline Result	%2.776	-

4.4. UCF101 Dataset

The UCF101 dataset includes more classes than other datasets. Most of the videos have been taken by one camera. Camera is mostly focused on one object or person which is the center of the action. Therefore, the appearance of different frames is similar. The difference is about the motion. Our methods could not learn that very well unless optical flow input is used. Inputting optical flow was not enough to learn motion relation between frames for optical flow CNN and optical flow feature concatenating CNN. However, optical flow modified recurrent CNN provided relatively good performance in the validation set. The performance of feature concatenating CNNs were below both standard networks and modified recurrent networks in their own categories. Forcing feature concatenating networks

to learn relations without having separate weights for learning relations causes drops in the performance in a lot of cases.

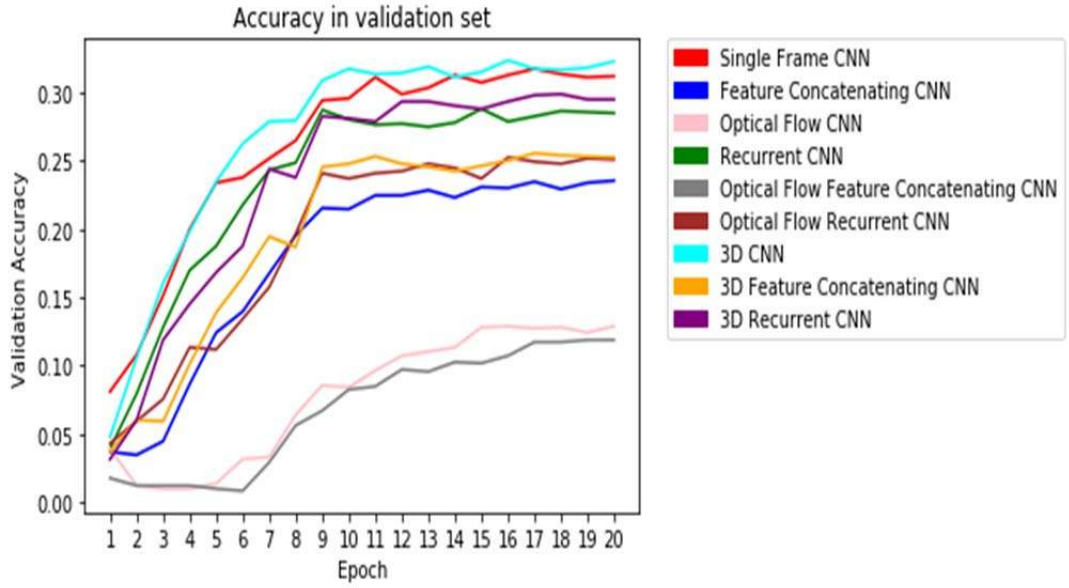

Figure 4.5. Validation Accuracies for the UCF101 dataset.

Modified recurrent networks almost always obtain lower losses even when they do not obtain better accuracies. In the UCF101 dataset, lowest losses are obtained by modified recurrent networks. For the testing set, obtained results are similar to the results of the validation set. Our proposed modified recurrent network can provide performance benefit in optical flow networks and two-stream networks. However, feature concatenating networks do not provide any performance benefit in any of the categories. Combining multiple networks provides the best performances for this dataset in our tests. By combining 6 networks (all networks except 3D networks), an accuracy of almost %33 is obtained.

Table 4.3. Accuracies in the test set of the UCF101 dataset.

Name of the Method	**Overall Accuracy**	**Accuracy Based on the Last Frame**
Single-Frame CNN	%25.399	%20.228
Feature Concatenating CNN	%21.673	%19.316
Modified Recurrent CNN	%23.992	%19.810
Optical Flow CNN	%13.688	%09.354
Optical Flow Feature Concatenating CNN	%12.243	%09.696
Optical Flow Modified Recurrent CNN	%24.296	%18.023
3D CNN	%25.095	%20.228
3D Feature Concatenating CNN	%21.483	%17.148
3D Modified Recurrent CNN	%24.373	%20.152
Two-Stream CNN	%26.122	%20.646
Two-Stream Feature Concatenating CNN	%25.361	%21.711
Two-Stream Modified Recurrent CNN	%29.810	%25.133
Combination of All Networks	%32.281	%29.201
Combination of All Networks Except 3D Networks	%32.585	%28.707
Baseline Result	%1.027	-

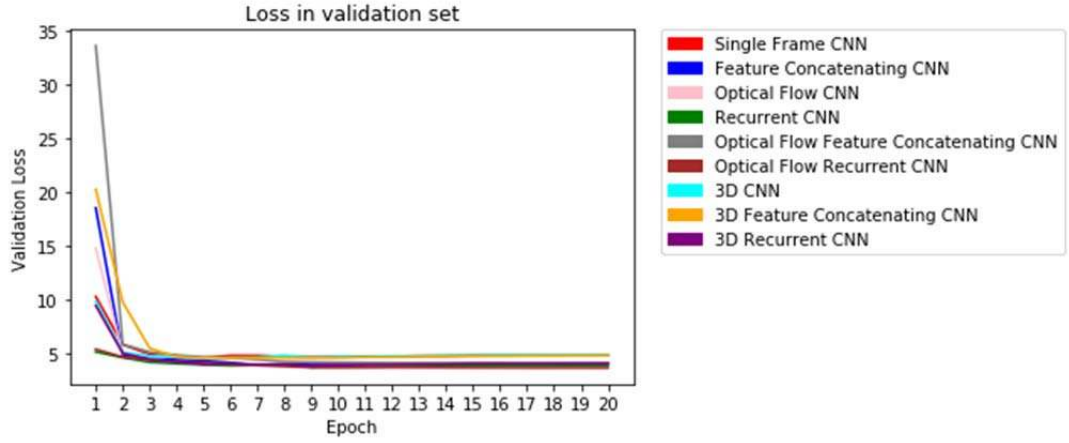

Figure 4.6. Validation Losses for the UCF101 dataset.

4.5. Using Only 2 Classes from the UCF-101 Dataset

The same networks have been trained and tested with a small portion of the UCF-101 dataset. Only two classes have been used from the UCF-101 dataset. Those classes are basketball dunk and volleyball spiking. Those two classes have been used because they are similar (both of them are indoor sports). This experiment has been carried out to evaluate the performance of the methods for small datasets and to prove that the network can provide better accuracies if larger networks are used. The networks have only been trained for 5 epochs.

For this subset of UCF-101 dataset, feature concatenating networks provide better accuracy while the accuracy of the modified recurrent networks drops. In all categories except optical flow networks, feature concatenating networks outperform standard networks while modified recurrent networks fall behind other networks in all categories. Unless a large dataset is used, modified recurrent CNN cannot learn very well to benefit from features coming from different frames. It requires to be trained for longer time and with more data. Feature concatenating convolutional neural networks apply convolution operation on the concatenated feature maps which does not require a longer training than training of standard convolutional neural networks. Normally, feature concatenating CNN is not as

good as modified recurrent CNN about using features from multiple frames. However, this task is simpler when the number of classes and samples are small.

Table 4.4. Accuracies for only 2 classes of the UCF-101 test set.

Name of the Method	**Overall Accuracy**	**Accuracy Based on the Last Frame**
Single-Frame CNN	%81.633	%83.673
Feature Concatenating CNN	%85.714	%85.714
Modified Recurrent CNN	%77.551	%73.469
Optical Flow CNN	%89.796	%73.469
Optical Flow Feature Concatenating CNN	%87.755	%65.306
Optical Flow Modified Recurrent CNN	%83.673	%67.347
3D CNN	%77.551	%79.592
3D Feature Concatenating CNN	%79.592	%83.673
3D Modified Recurrent CNN	%75.510	%73.469
Baseline Result	%48.98	-

5. CONCLUSION AND FUTURE WORK

Action recognition has been and will continue to be an active research area of computer vision. Deep learning has been a triggering factor for improvements in many areas of computer vision including action recognition. The main challenge of action recognition is using both spatial and temporal features. The scene and object information in the frames of the video is represented by spatial features. The relation between frames is represented by temporal features. Each method has its own way of learning and using features.

In this thesis, two methods are proposed. The main aim of the methods is detecting relations between frames by incorporating features from two frames. The first proposed method is named as recurrent CNN. It uses a separate convolution operation to combine features from two different frames in the same video. The second proposed method is named as feature concatenating CNN. It concatenates features from two different frames without using a separate convolution operation. The mechanism of feature concatenating CNN is not as sophisticated that of recurrent CNN.

We have used 3 datasets for testing. Those are Hollywood2, HMDB51 and UCF101. We have created 9 neural networks. 3 of them receive a single frame as input. 3 of them receive optical flow displacement fields. 3 of them receive a volume of frames to use 3D convolution operation. Networks receiving same type of input form a category. In each category, there is a standard convolutional network, a recurrent network and a feature concatenating network. Additionally, two-stream networks have been tested for each category. Those networks have been tested in 3 datasets. The obtained results indicate that the recurrent CNN is useful alternative to other methods. It can outperform standard convolutional networks in a lot of cases. However, feature concatenating convolutional neural networks fall behind standard convolutional neural networks most of the time. The performances depend on the dataset and the category of the network.

For future work, larger networks and better testing strategies can be used to obtain better performance. Currently, our networks have less filters and less layers than those of the studies that obtain competitive performances. Secondly, alternative ways can be explored to incorporate features from different frames.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P.A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., & Zhang, X. (2016). TensorFlow: A System for Large-Scale Machine Learning. *OSDI*.

Agarap, A.F. (2018). Deep Learning using Rectified Linear Units (ReLU). *ArXiv, abs/1803.08375*.

Arandjelovic, R., Gronát, P., Torii, A., Pajdla, T., & Sivic, J. (2016). NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. *CVPR*.

Carreira, J.F., & Zisserman, A. (2017). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4724-4733.

Castelluccio, M., Poggi, G., Sansone, C., Verdoliva, L. (2015). Land Use Classification in Remote Sensing Images by Convolutional Neural Networks. CoRR, abs/1508.00092.

Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. (2012). A brief introduction to OpenCV. *2012 Proceedings of the 35th International Convention MIPRO*, 1725-1730.

Diba, A., Sharma, V., & Gool, L.V. (2017). Deep Temporal Linear Encoding Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1541-1550.

Farnebäck, G. (2003). Two-Frame Motion Estimation Based on Polynomial Expansion. *SCIA*.

Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering, 9*.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv, abs/1502.03167*.

Kalogeiton, V., Weinzaepfel, P., Ferrari, V., & Schmid, C. (2017). Action Tubelet Detector for Spatio-Temporal Action Localization. *2017 IEEE International Conference on Computer Vision (ICCV)*, 4415-4423.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-Scale Video Classification with Convolutional Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1725-1732.

Kingma, D.P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *CoRR, abs/1412.6980*.

Kong, Y., & Fu, Y.R. (2018). Human Action Recognition and Prediction: A Survey. *ArXiv, abs/1806.11230*.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T.A., & Serre, T. (2011). HMDB: A large video database for human motion recognition. *2011 International Conference on Computer Vision*, 2556-2563.

Laarhoven, T.V. (2017). L2 Regularization versus Batch and Weight Normalization. *ArXiv, abs/1706.05350*.

Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B. (2008). Learning realistic human actions from movies. 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK/USA.

Liang, M., Hu, X. (June, 2015). Recurrent Convolutional Neural Network for Object Recognition. Paper presented at the CVPR2015. Boston, MA/USA.

Liu, J., Luo, J., & Shah, M. (2009). Recognizing realistic actions from videos "in the wild". *2009 IEEE Conference on Computer Vision and Pattern Recognition,* 1996-2003.

Marszalek, M., Laptev, I., Schmid, C. (2009). Actions in context. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2929-2936. Miami, FL/USA.

Sharma, S., Kiros, R., & Salakhutdinov, R.R. (2015). Action Recognition using Visual Attention. *ArXiv, abs/1511.04119*.

Simonyan, K., & Zisserman, A. (2014). Two-Stream Convolutional Networks for Action Recognition in Videos. *ArXiv, abs/1406.2199*.

Soomro, K., Zamir, A.R., & Shah, M. (2012). UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *ArXiv, abs/1212.0402*.

Tang, P., Wang, X., Shi, B., Bai, X., Liu, W., & Tu, Z. (2016). Deep FisherNet for Object Classification. *ArXiv, abs/1608.00182*.

Tran, D., Bourdev, L.D., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning Spatiotemporal Features with 3D Convolutional Networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, 4489-4497.

Walt, S.V., Colbert, S.C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering, 13*, 22-30.

Wang, L., Li, W., Li, W., & Gool, L.V. (2018). Appearance-and-Relation Networks for Video Classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1430-1439.

Werbos, P.J. (1990). Backpropagation Through Time: What It Does and How to Do It. Proceeding of the IEEE, 78(10), 1550 - 1560. doi:10.1109/5.58337.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R.R., Zemel, R.S., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *ICML*.

Zuo, Z., Shuai, B., Wang, G., Liu, X., Wang, X., Wang, B. (2016). Learning Contextual Dependence with Convolutional Hierarchical Recurrent Neural Networks. IEEE Transactions on Image Processing, 25, 2983-2996.

CURRICULUM VITAE

Mert Çopur was born in Adana, in 1994. He has graduated from Cebesoy Primary School in 2008 and 75. Yıl Anatolian High School in 2012. He got his BSc degree in computer engineering from the department of computer engineering in Cukurova University, in 2017.