Deep Learning for Recognition of Objects, Activities, Faces, and Spatio-temporal Patterns

by

AMIR GHADERI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2018

ACKNOWLEDGEMENTS

To my parents,

To my wife, and my son, Hamideh and Keon

To Soheil Shafiee who gifted me studying Copmuter Science

To Vassilis Athitsos

May, 2018

ABSTRACT

Deep Learning for Recognition of Objects, Activities, Faces, and Spatio-temporal Patterns

Amir Ghaderi, Ph.D.

The University of Texas at Arlington, 2018

Supervising Professor: Vassilis Athitsos

A popular method in machine learning is Convolutional Neural Network (CNN). CNN had was of high interest to the research community in the 1990s, but after that its popularity receded compared to the Support Vector Machine Support Vector Machine (SVM)[1]. One of the reasons was the relatively lower computational demands of SVM. Training CNNs requires significantly more computational power, time, and data than training SVM. One of the important issues in showing the power of the CNN is the availability of the huge amount of data and introducing big datasets. With increased availability of powerful GPU processing, using several improvements in network structure, and using much more data Krizhevsky et al. [2] used CNN to achieve the highest image classification accuracy on ImageNet Large Scale Visual Recognition Challenge(ILSVRC) [3]. After that result, CNNs have become widely popular in the computer vision and pattern recognition community, and have been applied to a variety of classification problems, including detection and localization. CNNs have achieved the best results for detection on the PASCAL VOC dataset [1], and for classification on the Caltech-256 [4] and Caltech-101 datasets [4, 5]. Based on such results, CNNs have emerged as a leading method for Machine learning and the term Deep Learning was emerged.

The origin of deep learning is in computer vision. However, researchers found that deep learning is a very powerful tool to solve many problems in other areas like forecasting, finance, human pose estimation, Natural Language Processing (NLP), etc. Deep learning based methods showed a wonderful performance relate to other available methods. We have tried to improve deep learning methods and using them for solving problems in different areas. In this thesis, we will try to use the deep learning techniques for solving problems in different areas such as unsupervised learning, object classification, forecasting, cognitive behavior assessment and face recognition.

In the computer vision part, a novel method for unsupervised feature learning for image classification was proposed in the thesis. Training CNN needs huge amount of data. So, finding the methods to train CNN with unlabeled data is very promising. In the second part, we proposed a new deep learning based framework for forecasting. Forecasting is a challenging task and has many applications in finance, meteorology, etc. We have proposed a new framework for forecasting in cases that there are many nodes to generate data. One application of our framework is prediction of the wind speed for multiple stations around the country. Another problem that we have been using Deep Learning (DL) to solve is face recognition at scale. Face recognition is very demanding both in academic and industry. We applied DL for solving face recognition for more than 600,000 identities. Also, we used DL to improve the performance of the system for behavioral assessment.

This thesis makes the following contributions. First, we proposed a method for unsupervised feature learning for object classification. Due to need for huge amount of labeled data for training neural networks, unsupervised learning is very appealing for CNN training. Representation learning with unlabeled data is an interesting and open problem in machine learning community. We used transfer learning to transfer knowledge from trained network in a dataset to test samples from other dataset. The results are promising and we

compare them to other methods. There are some ideas in this topic to improve the results which we implement them in the future. The paper was published at ICPR 2016.

Second, we solved a forecasting problem with proposing a new deep learning based framework. We presented a *spatio-temporal* wind speed forecasting algorithm using DL and in particular, Recurrent Neural Networks (RNNs). we modeled the spatio-temporal information by a graph whose nodes are data generating entities and its edges basically model how these nodes are interacting with each other. Available methods for forecasting propose models to forecast wind speed for only one node. One of the main contributions of our work is the fact that we obtain forecasts of all nodes of the graph at the same time based on one framework. Our paper in this project was published at ICML Time Series workshop 2017.

We improved the motion analysis module for HTKS assessment. HTKS [6] is a game-like cognitive assessment method, designed for children between four and eight years of age. During the HTKS assessment, a child responds to a sequence of requests, such as "touch your head" or "touch your toes". The cognitive challenge stems from the fact that the children are instructed to interpret these requests not literally, but by touching a different body part than the one stated. In prior work, we have developed the CogniLearn system, that captures data from subjects performing the HTKS game, and analyzes the motion of the subjects. We propose specific improvements that make the motion analysis module more accurate. As a result of these improvements, the accuracy in recognizing cases where subjects touch their toes has gone from 76.46% in our previous work to 97.19%. The paper was published at PETRA 2017.

Finally, a method proposed for face recognition at scale for large number of identities. We used the triplet loss function to train the neural network for feature learning. In our problem for face recognition we have huge number of classes so we can not use softmax in the last layer of the network like what is done for usual classification problems. So, we

used the triplet loss function for the network to create features and then we used a classifier on top of the features. The triplet loss function tries to minimize the distance of samples in a class and maximize the distance of a class with other classes. As a result of CNN for representation learning, each image could be converted to a 128-dimensional vector. We have done experiments on different number of classes on different datasets like FLW, Mega Face, and Face Scrub. The number of classes are 500, 5K, 10K, 20K, 100K, and 663386.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

A popular method in machine learning is CNN. CNN had was of high interest to the research community in the 1990s, but after that its popularity receded compared to the Support Vector Machine SVM[1]. One of the reasons was the relatively lower computational demands of SVM. Training CNNs requires significantly more computational power, time, and data than training SVM. One of the important issues in showing the power of the CNN is the availability of the huge amount of data and introducing big datasets. With increased availability of powerful GPU processing, using several improvements in network structure, and using much more data Krizhevsky et al. [2] used CNN to achieve the highest image classification accuracy on ImageNet Large Scale Visual Recognition Challenge(ILSVRC) [3]. After that result, CNNs have become widely popular in the computer vision and pattern recognition community, and have been applied to a variety of classification problems, including detection and localization. CNNs have achieved the best results for detection on the PASCAL VOC dataset [1], and for classification on the Caltech-256 [4] and Caltech-101 datasets [4, 5]. Based on such results, CNNs have emerged as a leading method for Machine learning and the term Deep Learning was emerged.

The origin of deep learning is in computer vision. However, researchers found that deep learning is a very powerful tool to solve many problems in other areas like forecasting, finance, human pose estimation, NLP, etc. Deep learning based methods showed a wonderful performance relate to other available methods. We have tried to improve deep learning methods and using them for solving problems in different areas. In this thesis, we will try to use the deep learning techniques for solving problems in different areas such as unsupervised learning, object classification, forecasting, cognitive behavior assessment and face recognition.

In the computer vision part, a novel method for unsupervised feature learning for image classification was proposed in the thesis. Training CNN needs huge amount of data. So, finding the methods to train CNN with unlabeled data is very promising. In the second part, we proposed a new deep learning based framework for forecasting. Forecasting is a challenging task and has many applications in finance, meteorology, etc. We have proposed a new framework for forecasting in cases that there are many nodes to generate data. One application of our framework is prediction of the wind speed for multiple stations around the country. Another problem that we have been using DL to solve is face recognition at scale. Face recognition is very demanding both in academic and industry. We applied DL for solving face recognition for more than 600,000 identities. Also, we used DL to improve the performance of the system for behavioral assessment. In the following paragraphs, we explain more about each section of the thesis.

Due to need for huge amount of labeled data for training neural networks, unsupervised learning is very appealing for CNN training. Representation learning with unlabeled data is an interesting and open problem in machine learning community. In the chapter 2, we propose a method for unsupervised feature learning for object classification. To show the capability of the proposed method we have done experiments in popular datasets in this area like STL-10, CIFAR-10, and CIFAR-100. We used transfer learning to transfer knowledge from trained network in a dataset to test samples from other dataset. The results are promising and we compare them to other methods. There are some ideas in this topic to improve the results which we implement them in the future.

In the chapter 3 we want to solve a forecasting problem with deep learning techniques. We present a *spatio-temporal* wind speed forecasting algorithm using DL and in particular, RNNs. Motivated by recent advances in renewable energy integration and smart grids, we apply our proposed algorithm for wind speed forecasting. Renewable energy resources (wind and solar) are random in nature and, thus, their integration is facilitated with accurate short-term forecasts. The problem is to forecast the wind speed for different measuring stations. In our proposed framework, we model the spatio-temporal information by a graph whose nodes are data generating entities and its edges basically model how these nodes are interacting with each other. Available methods for forecasting propose models to forecast wind speed for only one node. One of the main contributions of our work is the fact that we obtain forecasts of all nodes of the graph at the same time based on one framework.

We report the results of a case study on recorded time series data from a collection of wind mills in the north-east of the U.S. The goal is to show that the proposed DL-based forecasting algorithm significantly improves the short-term forecasts compared to a set of widely-used benchmarks models. Our paper in this project was published at ICML Time Series workshop 2017.

In the chapter 4, we are trying to improve the motion analysis module for HTKS assessment. HTKS [6] is a game-like cognitive assessment method, designed for children between four and eight years of age. During the HTKS assessment, a child responds to a sequence of requests, such as "touch your head" or "touch your toes". The cognitive challenge stems from the fact that the children are instructed to interpret these requests not literally, but by touching a different body part than the one stated. In prior work, we have developed the CogniLearn system, that captures data from subjects performing the HTKS game, and analyzes the motion of the subjects. We propose specific improvements that make the motion analysis module more accurate. As a result of these improvements, the accuracy in recognizing cases where subjects touch their toes has gone from 76.46% in our previous work to 97.19%.

The purpose of the chapter 5 is to propose a method for face recognition at scale for large number of identities. There are many datasets for face recognition and details about them are mentioned in the 5.3. One of the recently published datasets is Mega Face dataset with 672,057 identities and more than 4.7 million images. Our goal is to propose a end to end method for face recognition in the dataset. We used the triplet loss function to train the neural network for feature learning. In our problem for face recognition we have huge number of classes so we can not use softmax in the last layer of the network like what is done for usual classification problems. So we used the triplet loss function for the network to create features and then we used a classifier on top of the features. The triplet loss function tries to minimize the distance of samples in a class and maximize the distance of a class with other classes. As a result of CNN for representation learning, each image could be converted to a 128-dimensional vector.

We have done experiments on different number of classes on different datasets like FLW, Mega Face, and Face Scrub. The number of classes are 500, 5K, 10K, 20K, 100K, and 663386 so far. As a plan for future works we will try to all distractors of the Mega Face dataset to the test set and make improvement in feature extraction part.

1.1 Contributions

This thesis investigates improvements in application of deep learning in many areas such as unsupervised learning, forecasting, and cognitive behavior assessment. Our contributions are as follow:

- Proposing a new method for unsupervised feature learning
- Running extensive amount of experiments with Caffe framework with different hyper parameters and apply transfer learning to trained models for showing power of our method for unsupervised feature learning
- Designing and implementing a novel framework for forecasting to predict output values for all nodes in a graph

- Improvement in vision module of CogniLearn system for Cognitive Behavior Assessment
- Proposing a framework for face recognition for more than 650K individuals

1.2 Published papers

As a result of our researches, some papers were published in different venues. The published papers are as follows

- Deep Forecast: Deep Learning-based Spatio-Temporal Forecasting, A Ghaderi, BM Sanandaji, F Ghaderi, The 34th International Conference on Machine Learning (ICML), Time series Workshop, 2017
- Scalable Deep Traffic Flow Neural Networks for Urban Traffic Congestion Prediction, M Fouladgar, M Parchami, R Elmasri, A Ghaderi, International Joint Conference on Neural Networks (IJCNN), 2017
- Improving the Accuracy of the CogniLearn System for Cognitive Behavior Assessment, A Ghaderi, S Gattupalli, D Ebert, A Sharifara, V Athitsos, F Makedon, Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA), 2017
- Selective Unsupervised Feature Learning with Convolutional Neural Network (S-CNN), Amir Ghaderi, Vassilis Athitsos, International Conference on Pattern Recognition (ICPR), 2016
- Evaluation of Deep Learning based Pose Estimation for Sign Language Recognition, Srujana Gattupalli, Amir Ghaderi, Vassilis Athitsos, Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA), 2016

1.3 Future works

1. Unsupervised Feature Learning:
   (a) Training our model on data from the ImageNet dataset
   (b) Running the experiments with bigger network structures
2. Forecasting:
   (a) Trying to report the results of our model on different datasets
   (b) Testing the model in different problems which can modeled as a graph like traffic prediction
3. Cognitive Behavior Assessment:
   (a) collecting data from children between the ages of 4-8.
   (b) Using the depth modality in addition to the color modality
   (c) Identifying "Self-correction" cases, where the subject starts moving hands towards the wrong part, but then self-corrects the motion.
4. Face Recognition at Scale:
   (a) Adding distractors of the Mega Face dataset to the test set
   (b) Improving the accuracy of the face recognition by different structure for features extraction
   (c) Creating a method for proposing many classes with their probabilities for each test image instead of one class
   (d) Finding a way to predict the performance of a method on N classes based on performance on N/5 classes

CHAPTER 2

Selective Convolutional Neural Networks for Feature Learning

A popular method in machine learning is CNN. CNN had was of high interest to the research community in the 1990s, but after that its popularity receded compared to the Support Vector Machine SVM[1]. One of the reasons was the relatively lower computational demands of SVM. Training CNNs requires significantly more computational power, time, and data than training SVM. One of the important issues in showing the power of the CNN is the availability of the huge amount of data and introducing big datasets. With increased availability of powerful GPU processing, using several improvements in network structure, and using much more data Krizhevsky et al. [2] used CNN to achieve the highest image classification accuracy on ImageNet Large Scale Visual Recognition Challenge(ILSVRC) [3]. After that result, CNNs have become widely popular in the computer vision and pattern recognition community, and have been applied to a variety of classification problems, including detection and localization. CNNs have achieved the best results for detection on the PASCAL VOC dataset [1], and for classification on the Caltech-256 [4] and Caltech-101 datasets [4, 5]. Based on such results, CNNs have emerged as a leading method for Machine learning and the term Deep Learning was emerged.

2.1 Introduction

Deep learning could solve many problems which is hard to solve by other traditional methods in machine learning. The origin of deep learning is in computer vision but the application of deep learning is increasing to many other areas like forecasting, finance, etc. At the same time, a weakness of supervised learning using CNN is the need for much larger amounts of labeled training data, compared to alternative methods. Acquiring a large number of labeled instances requires oftentimes significant time spent by humans to provide the labels, and significant costs. Furthermore, when training instances are labeled by humans, errors and inconsistency in labeling become an issue, especially when labeling large scale datasets. On the other hand, in many settings it is easy to obtain vast amounts of unlabeled data, making unsupervised learning an attractive alternative, provided of course that unsupervised learning can attain satisfactory accuracy.

The problem is find a way to train the CNN with unlabeled data and train the features from data with no labels for object classification. As we mentioned earlier, annotating large sets of images can be an important bottleneck for training supervised methods, but large amounts of unlabeled data may be easy to obtain. E.g. in the STL dataset there are 100K unlabeled images. We propose an algorithm that learns features using CNNs that train on unlabeled data. The evaluations of the algorithm on the STL, CIFAR-10, CIFAR-100 datasets are done. The results shows competitive performance compared to other methods.

2.2 Related works

CNN typically consist of different types of layers, with each layer performing some specialized functionality. Examples of such types of layers are convolutional layers, rectifier layers(max(0,x)) (also known as ReLU layers), max-pooling layers for reducing the number of inputs, and normalization layers [2]. The speed of training Deep CNN with ReLUs is much higher than the speed of training Deep CNN with tanh units [2]. In fully connected layers, each element is calculated based on the values of all components of the input. The last layer calculates the loss function of the network. The main role of training is on the convolutional layers, and classification is performed by the fully connected layers. After training a CNN, instead of performing classification using the fully connected layers, one can feed features from the last convolution layer into an SVM classifier.

CNNs can be combined with both supervised and unsupervised methods in an end-to-end system. In supervised methods, data augmentation can be used to increase the number of instances for training, so as to reduce overfitting. Coates et al. [10] point out that the effect of certain factors, such as the number of hidden nodes, may be more vital for performance than the depth of the model. In [11], researchers use the temporal slowness constraint with and employ a linear autoencoder in order to learn features from video. In the category of unsupervised methods, Bo et al. propose the hierarchical matching pursuit (HMP) method, which uses sparse coding and learns hierarchical feature representations in an unsupervised manner on depth data [12]. Unsupervised feature learning is used by Netzer et al. for recognizing digits cropped from street view images[13]. Features invariant to transformations are learned by Sohn et al. [14]. Le et al. [15] have trained features robust to translation, scaling, and rotation for face detection using a deep sparse auto encoder on a large dataset, without having to label images.

2.3 Our Method

Object detection in many methods is based on exhaustive search for specific object types. Alternatively, some methods output possible locations of objects, without being trained to detect specific types of objects. Such methods include objectness[16], selective search [17], and category-independent object proposals[18].

Selective search identifies potential object locations which can be used for object recognition. It combines advantages of both exhaustive search and segmentation and achieves relatively high speed compared to alternative methods. It uses the structure of the image for sampling, and it creates scores by merging low-level superpixels. The goal of

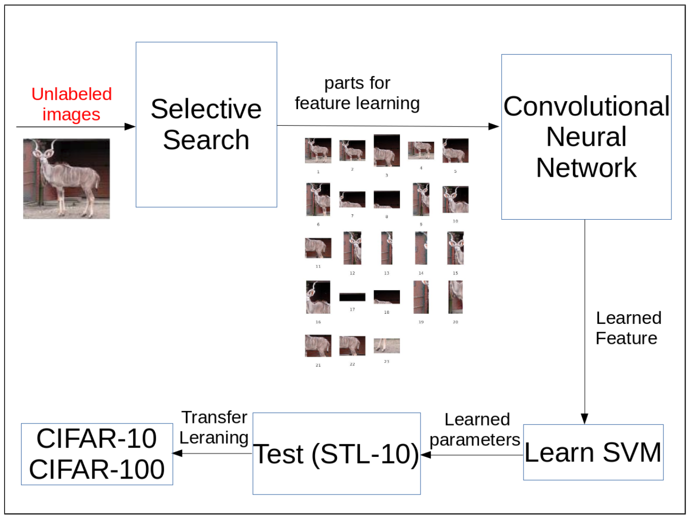

Figure 2.1: Overview of the SCNN algorithm

selective search is to find all locations in the image that have high probability to be an object. The output of selective search given an image is a set of bounding boxes, representing possible locations of objects.

Let x_i be an unlabeled image, that we give as input to the selective search algorithm. Selective search outputs a set w_i of bounding boxes for x_i. We treat each bounding box as a subimage of x_i. Thus, set w_i consists of many images a_{ij}, which are all subimages of x_i.

$$w_i = a_{ij} | a_{ij} \text{ is output of selective search with input} x_i \tag{2.1}$$

If selective search creates T_i subimages from x_i then $j = 1, 2, 3....T_i$ and $w_i = a_{i1}, a_{i2}, ...a_{iT_i}$. Then, we assign training label i to all these images in set w_i . In other words,w_i generates T_i image label pairs $[a_{ij}, i]$ for our training set. Intuitively, all subimages from the same original imagex_i are assigned i as their training label. Thus, training labels are assigned fully automatically, with no need for manual intervention. Set T contains as elements the numbers T_i of subimages extracted from all unlabeled images x_i. We have 100,000 unlabeled images in the STL dataset, so T has 100,000 members.

$$T = T_1, T_2, ..., T_{100000} \tag{2.2}$$

Suppose that we want to train a CNN to recognize C classes, where C is a user-specified parameter. We want to find the C members of T that contain the most elements. For reaching this goal we sort set T in descending order, and we put the indices in set TS.

$$TS = \text{indices of sorted T in descending order} = ts_1, ts_2, ...ts_{100000} \tag{2.3}$$

Note that TS stores indices of elements in T, not the elements themselves. So T_{ts_1} is the maximum element of the T. We choose the top C indices of TS to train the CNN. In our experiments, we try C= 5000, 10000, 15000, 20000, 25000, 30000. Our goal is to train a CNN to discriminate between C classes, and to choose features that can discriminate among various types of objects. Therefore, the input for training the CNN is a set of images X and labels as below:

$X = w_{ts_1}, w_{ts_2}, ..., w_{ts_C}$

labels for images in $w_{ts_i} = ts_i$

The loss function which should be minimized is l(i,a_{ij}) is the softmax loss based on the image a_{ij} and the label i. In the following section we provide more details about the architecture of our CNN.

In Figure 1 we show the overview of proposed algorithm. Selective search finds the important parts of the object. Then CNN learns the features to classify those important parts. For classification, we use the SVM instead of the fully connected layers in the network. At the final step, an SVM is trained on the features.

2.4 Experiments

For comparison to other methods, we evaluate performance on the STL-10 dataset [10], which has 10 classes, and the CIFAR-10, and CIFAR-100 datasets [19] that have 10 and 100 classes respectively. STL-10 contains 100,000 unlabeled data we use it as source of data for unsupervised feature learning. We extract the surrogate classes for training the CNN from the unlabeled set of STL-10. Each image in the unlabeled STL set is given an input for selective search. The output images of the selective search have different sizes, which would cause features created in fully connected layers to have different numbers of elements. To deal with this problem there are two options. The first is resizing the images to P*P fixed size, where P is a preselected parameter. The second is to use images with different sizes at beginning of the network, and to use spatial pyramid pooling [18] at the last layer before the fully connected layers, so as to create fixed number of features in fully connected layers. Here we select the first option and resize the input images to 32*32.

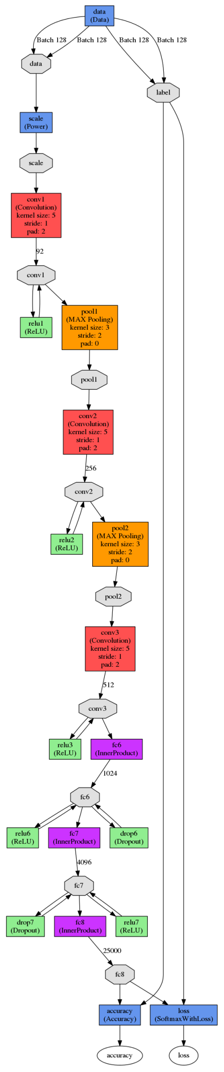

Figure 2.2: Overview of the SCNN archetecture

We try two network architectures. The first one has three convolutional layers, each of them with 64, 128, and 256 filters respectively. The kernel size for the first convolutional layer is 5*5. We use stride 1 and padding 2 for this layer. An ReLU filter is after each convolutional layer. After the first and the second ReLU layer we have the max pooling layer. Here we have kernel size 3*3, stride 2, and zero padding. The third ReLU layer is followed by two fully connected layers with 512 and C neurons respectively, where C is the number of the class labels that are assigned automatically. Note that C varies in different experiments, as described later. Dropout is employed at the fully connected layers to reduce overfitting. At the end there is a softmax layer for calculating the loss function. We named this network 64 – 128 – 256_512.

The second network, which is larger than the first one, has three convolutional layers with 92, 256, and 512 filters, followed by a fully connected layer with 1024 neurons. We named this network 92 – 256 – 512_1024. The kernel size for the first convolutional layer is 5*5. We use stride 1 and padding 2 for this layer. Again, a Rectified Linear Unit (ReLU) is used after each convolutional layer. After the first ReLU layer there is a max pooling layer with kernel size 3*3, stride 2, and zero padding.

The second convolutional layer is like the first one, except that it consists of 256 kernels instead of 92. The ReLU and pooling layers applied to second convolutional layer are the same as for the first layer. The third convolutional layer has 512 kernels. At the end we have two fully connected layers with 1024 and C neurons, where again C is the number of classes and is different in each experiment. As in the first network, we have a softmax layer at the end for calculating the loss function. Figure 2.2 shows the second network in details.The figure is created by NVIDIA Deep Learning GPU Training System (DIGITS). We implement CNNs based on the caffe framework. For each dataset, each image of the test set of that dataset is given as input to the network. Then, we compute the output of all the network layers expect the top softmax one.

We use the features for training a one-vs-all linear support vector machine (SVM). To train the SVM we use the standard training and testing protocols for each dataset. For the STL dataset, we use the 10 predefined folds for training the SVM, and final accuracy is calculated as the average accuracy over the 10 splits. Here we investigate the impact of different parameters on the results. We run different experiments by varying the number of classes, the network structure, and the dataset.

#Classes	CNN	SVM
5000	64-128-256_512	58.01
10000	64-128-256_512	58.10
15000	64-128-256_512	58.29
20000	64-128-256_512	61.04
25000	64-128-256_512	60.38
30000	64-128-256_512	58.87

Table 2.1: Accuracy percentages on the STL dataset using different values of C (number of classes).

Architecture	#classes	Accuracy
64-128-256_512	20000	61.04
64-128-256_512	25000	60.38
92-256-512_1024	20000	60.36
92-256-512_1024	20000	61.94

Table 2.2: Accuracy percentages of different architectures on the STL dataset.

2.4.1 Number of classes

The Parameter C is the number of classes that are assigned in an automatic manner, so as to train the CNN. We experimented with C equal to 5K, 10k, 15K, 20K, 25K, and 30K. A larger C can increase accuracy, because the neural network receives more training data. At the same time, when C is too large, the network can be fed with conflicting data (since class labels are assigned automatically) and not converge. Table 2.1 shows the accuracy obtained on the STL dataset for different values of C.

2.4.2 Generality of features

We have also used the features learned on the STL dataset for recognition on the CIFAR-10 and CIFAR-100 datasets. Both datasets are split into a training set and a test set. In contrast to the STL dataset, the CIFAR datasets do not have any unlabeled data. We do not use their training set to learn features by CNN, using instead the trained features from the STL-10 dataset. The results are comparable to other methods which use the CIFAR training sets directly. Table 2.3 shows the results for classification on CIFAR-10 and CIFAR-100 with learned features from STL-10.

Method	CIFAR-10	CIFAR-100
64-128-256_512	72.68	47.70
92-256-512_1024	75.17	51.27
[10]	79.7	70.2
[20]	-	54.32

Table 2.3: Classification accuracy percentages on the CIFAR-10 and CIFAR-100 datasets

S-CNN	[21]	[11]	[22]	[14]	[10]
61.94	70.10	61.0	60.1	58.7	51.5

Table 2.4: Classification accuracy percentages on the STL-10 dataset

2.4.3 Different network architectures

Since we established the best range for parameter C (number of classes) is 20K-25K, we decided to run two different architectures for the neural network, trained with C equal to 20000 and 25000. The 92-128-512_1024 network has more parameters to learn and more power to discriminate between classes relative to the 64-128-256_512 network. We only change the parameters of the layers, and the number of layers is fixed for both network architectures. The 64-128-512_1024 network with 25K classes has 61.94 percent accuracy on STL test set. It shows that this architecture has more power for creating more distinguishing features. Classification accuracy improves with increasing network size. This is evidence that our algorithm works well with larger networks and avoids overfitting. The results of these experiments with different neural network architecture on the STL-10 dataset are shown in table 2.2.

2.4.4 Comparison to other methods

In Table 2.4 we compare the results of our algorithm with other learning methods on the STL-10 dataset. Our approach appears to be competitive with the others, despite the fact that our model only uses 3 convolutional layers and requires learning only few parameters. Note that better result than ours which reported in the table have been obtained by using external data, achieving an accuracy rate of 70.10 on STL-10 . In that work, knowledge gained from previous optimizations is transferred to new tasks in order to find optimal hyperparameter settings more efficiently. We find it particularly promising that our results are more accurate than those of [11], [22], [14], and [10].

2.5 Further works

We want to propose a new method for unsupervised feature learning, tailored for image classification in large datasets. We showed that results are compatible to previously proposed methods, while our results use a simpler architecture and no data augmentation or use of external data. Also, the features learned on the STL-10 dataset are tested on the CIFAR-10 and CIFAR-100 datasets, and results show that the learned features generalize well and can extend to other sets of data. Our plan for future work is as follows

- Trying bigger and deeper architectures for CNN. Using CNNs with more layers may learn more powerful features for distinguishing among different objects.
- To try learning features from a bigger dataset, with more images and classes, to see if that would lead to learn better features.
- Using fc layers as classifier and comparing the results with the SVM classifier

2.6 References

R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in CVPR, 2014.

A. Krizhevsky, I. Sutskever and G. Hinton, "Imagenet classification with deep convolutional neural networks," in NIPS, 2012.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," IJCV, vol. 115, no. 3, pp. 211-252, 2015.

S. Gattupalli, A. Ghaderi and V. Athitsos, "Evaluation of Deep Learning based Pose Estimation for Sign Language Recognition," eprint arXiv:1602.09065, 2016.

M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in ECCV, 2014.

J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," arXiv preprint arXiv:1310.1531, 2013.

A. Coates, H. Lee and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in AISTATS , 2011.

W. Zou, A. Ng, S. Zhu and K. Yu, "Deep learning of invariant features via simulated fixations in video," in NIPS, 2012.

L. Bo, X. Ren and D. Fox, "Unsupervised Feature Learning for RGB-D Based Object Recognition," in International Symposium on Experimental Robotics (ISER), 2012.

K. Sohn and H. Lee, "Learning invariant representation with local transformations," in ICML, 2012. Q. V. Le, M. A. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean and A. Y. Ng, "Building high-level features using large scale unsupervised learning," in International Conference on Machine Learning, 2012.

B. Alexe, T. Deselaers and V. Ferrari, "Measuring the objectness of image windows," in TPAMI, 2012. J. Uijlings, K. Van de Sande, T. Gevers and A. Smeulders, "Selective search for object recognition," IJCV, vol. 104, no. 2, pp. 154-171, 2013.

I. Endres and D. Hoiem, "Category independent object proposals," in ECCV, 2010. K. Van de Sande, J. Uijlings, T. Gevers and A. Smeulders, "Segmentation as selective search for object recognition," ICCV, 2011.

j. Hosang, R. Benenson and B. Schiele, "How good are detection proposals, really?," in British Machine Vision Conference (BMVC), 2014.

A. Coates, A. Ng and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in AISTATS, 2011.

A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.

K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in TPAMI, 2015.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, " Caffe: Convolutional architecture for fast feature embedding," in ACM International Conference on Multimedia, 2014.

A. Dosovitskiy, J. T. Springenberg, M. Riedmiller and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," NIPS, 2014.

Y. Jia, C. Huang and T. Darrell, "Beyond Spatial Pyramids: Receptive Field Learning for Pooled Image Features," in CVPR, 2012.

K. Swersky, J. Snoek and R. Adams, "Multi-task bayesian optimization," in NIPS, 2013.

J. Mairal, P. Koniusz, Z. Harchaoui and C. Schmid, "Convolutional Kernel Networks," in NIPS, 2014.

A. Coates and A. Ng, "Selecting receptive fields in deep networks," in NIPS, 2011.

K. Sohn and H. Lee, "Learning invariant representations," in ICML, 2012.

P. Vincent, H. Larochelle, Y. Bengio and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in ICML, 2008.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu and A. Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," in NIPS, 2011.

The results of this work was published at ICPR 2016. "Selective Unsupervised Feature Learning with Convolutional Neural Network (S-CNN)", Amir Ghaderi, Vassilis Athitsos.

CHAPTER 3

Deep Forecast: Deep Learning-based Spatio-Temporal Forecasting

3.1 Introduction and problem definition

This chapter tries to use deep learning for solving the forecasting problem. We present a *spatio-temporal* wind speed forecasting algorithm using DL and in particular, RNNs. Motivated by recent advances in renewable energy integration and smart grids, we apply our proposed algorithm for wind speed forecasting. Renewable energy resources (wind and solar) are random in nature and, thus, their integration is facilitated with accurate short-term forecasts. In our proposed framework, we model the spatio-temporal information by a graph whose nodes are data generating entities and its edges basically model how these nodes are interacting with each other. One of the main contributions of our work is the fact that we obtain forecasts of all nodes of the graph at the same time based on one framework. We report the results of a case study on recorded time series data from a collection of wind mills in the north-east of the U.S. The goal is to show that the proposed DL-based forecasting algorithm significantly improves the short-term forecasts compared to a set of widely-used benchmarks models.

Many countries in the world and many states in the U.S. have mandated aggressive Renewable Portfolio Standards (RPSs). Among different renewable energy resources, wind energy itself is expected to grow to provide between 15 to 25% of the world's global electricity by 2050. One of the most important points of the problem is that we do not know the relationship between stations. We have to design a model that determines which stations are more important to forecast one specific station.

This work was published at ICML Time Series workshop 2017.

3.2 Related works

Wind speed forecasting methods can be categorized to different groups: (i) model-based methods such as Numerical Weather Prediction (NWP) vs. data-driven methods, (ii) point forecasting vs. probabilistic forecasting, and (iii) short-term forecasting vs. long-term forecasting. This paper is concerned with short-term point forecasting using both temporal data as well as spatial information. For a more complete survey of wind speed forecasting methods see [23] and [24], among others.

There is a growing interest in the so-called *spatio-temporal* forecasting methods that use information from neighboring stations to improve the forecasts of a target station, since there is a significant cross-correlation between the time series data of a target station and its surrounding stations. We review some of the spatio-temporal forecasting methods. [25] introduced the Regime Switching Space-Time Diurnal (RSTD) model for average wind speed data based on both spatial and temporal information. This method was later improved by Hering and Genton [26] who incorporated wind direction in the forecasting process by introducing Trigonometric Direction Diurnal (TDD) model. [27] also considered probabilistic TDD forecast for power system economic dispatch. [28] employed a multi-channel adaptive filter to predict the wind speed and direction by taking advantages of spatial correlations at numerous geographical sites. [29] presented Markov chain-based stochastic models for predictions of wind power generation after characterizing the statistical distribution of aggregate power with a graph learning-based spatio-temporal analysis. Regime-switching models based on wind direction are studied by [30] where they consider various statistical models, such as ARX models, to understand the effects of different variables on forecast error characteristics. A methodology with probabilistic wind power forecasts in the form of predictive densities taking the spatial information into account was developed in [31]. Sparse Gaussian Conditional Random Fields (CRFs) have also been deployed for probabilistic wind power forecasting [32].

3.2.1 Forecasting using Neural Networks

Among different DL algorithms, RNN has been commonly used in forecasting applications. [33] used Long Short Term Memory (LSTM) in text generation and predict one output in each time step. He used input values till time t to get prediction in time $t + 1$. [34] used a mixture of wavelet transform, deep belief network, and spine quantile regression for wind speed forecasting. [35] proposed a hybrid model using deep neural network. [36] used fuzzy logic and neural networks for forecasting. [37] proposed a comparison on three neural networks for 1-hour wind speed forecasting.

3.3 Recurrent Neural Networks and LSTM

Originating from computer vision and image classification, DL has shown promising results in different tasks in recent years [38], [39], [40]. Its ability in handling large amount of data and learning nonlinear and complicated models has made it an appealing framework. In one of the earliest works, [38] proposed to run a deep (a neural network with several hidden layers) CNN on a Graphics Processing Unit (GPU) to classify a large data set of images (ImageNet dataset, [41]). Among several algorithms that have been proposed in DL for different tasks, RNN is proposed for modeling temporal data and has been applied to speech recognition, activity recognition, NLP, etc. In the following, we provide some insights on how an RNN is built. Let $\mathcal{X} \triangleq \{x_1, x_2, \ldots, x_\ell\}$ be a sequence of data where x_t is the vector of features at time t and ℓ is the input horizon. There exist many variations for the RNN structure. Some structures generate output for each time step while there are RNNs with one final output at time ℓ when $\mathcal{X}$ is applied as an input to the RNN. Let $\mathcal{Y} \triangleq \{y_1, y_2, \ldots, y_\ell\}$ be the sequence of outputs at each time step. A function f is applied on each input x and the output of f in the previous time step. One should note that the same function should be used during all time steps. This is an important point which makes the model capture the useful information content of the data (used for training) at each time step.Stochastic Gradient Descent (SGD) and Back Propagation (BP) are used to train the function and find optimal parameters.

There exist some issues with the basic RNN structure such as vanishing gradient (especially for long input sequences). [42] proposed LSTM to address such problems. In short, LSTM provides a framework to embed the required information for the function. LSTM networks have better convergence performance compared to the basic RNN. LSTM consists of multiple functions as compared to one function in vanilla RNN. These functions try to remember the helpful and forget the unnecessary information from inputs. Figure 3.1 shows relationship between functions in LSTM. The output of each step is calculated following the formulas provided in (3.1):

$$\begin{aligned}
f_t &= g(W_f.x_t + U_f.h_{t-1} + b_f) \\
i_t &= g(W_i.x_t + U_i.h_{t-1} + b_i) \\
k_t &= tanh(W_k.x_t + U_k.h_{t-1} + b_k) \\
c_t &= f_t \times c_{t-1} + i_t \times k_t \\
o_t &= g(W_o.x_t + U_o.h_{t-1} + b_o) \\
h_t &= o_t \times tanh(c_t)
\end{aligned} \tag{3.1}$$

Where x_t is the input vector at time t and g is an activation function like $Sigmoid$ or $ReLU$. W, U are weight matrices and b is the bias vector. h_t and c_t are output and cell state vector at time t. f_t has served for remembering old information and i_t has served for getting new information.There are many variations of LSTM. Keen readers can find more about LSTM in [43].

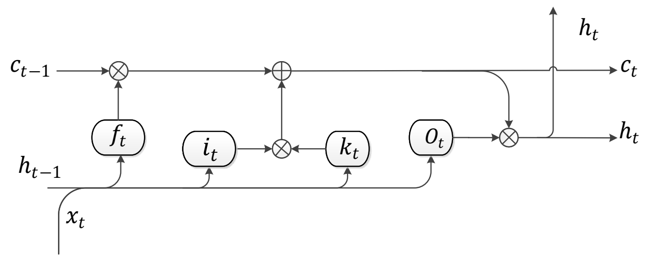

Figure 3.1: LSTM block at time t

3.4 DL-based Spatio-Temporal Forecasting (DL-STF)

In this section, we outline spatio-temporal forecasting scheme which is based on DL. We namely call our algorithm DL-STF. Let graph G be defined as $G \triangleq [E, V]$ where E denotes the edges and $V = \{v_1, \ldots, v_n\}$ denotes the nodes. Each node of the graph v_i generates data at each time step. Node v_i at time t generates x_i^t which is a scalar (e.g., wind speed in our problem). Assume x_i^t is sampled from an unknown distribution $P_{real}(x)$. Also $s^t = [x_1^t, x_2^t \ldots x_n^t]$ (it is a vector) contains the output of all nodes at time t. Similarly, let $\hat{x}_i^t$ be the output prediction of node i at time t. Let $\hat{s}^t = [\hat{x}_1^t, \hat{x}_2^t \ldots \hat{x}_n^t]$ be the vector containing prediction of all nodes (same size as s^t). We assume we only have real data for all nodes every h time steps. Our goal is to predict s^t, using $\{s^k\}, k \in \{t - \ell, t - \ell + 1, ..., t - 1\}$. Based on moving horizon scheme, when real value for s^k is not available, we use its prediction, $\hat{s}^k$.

3.4.1 Time step models

We have access to real data every h hours and want to forecast wind speed for the next h hours. In different time steps we have different kind of inputs. For the first time step, we have real data for all inputs but for the next time step, we have real data for all inputs except one. For that one we use forecast data from previous step. This scheme repeats for all the h steps. Based on this paradigm, we define a specific model for each time step over the input horizon. In order to train the model, we use real values as much as we can, but if we do not have real values, we use forecast values from previous trained models. So for example the model for forecasting at time $t + 2$ should differ from model for forecasting at time $t + 3$. If we have n stations, the input for our algorithm is an n-dimensional vector at each time step and we forecast an n-dimensional vector for next time step.

Let t denote the global time index, h the number of time steps in moving horizon, and ℓ the input horizon. We train h different models. Model number i is represented by M_i. We define $\hat{t} = t \bmod h$ and the relation between $\hat{t}$ and i is as follows:

$$i = \begin{cases} \hat{t} & ,\text{if } \hat{t} \neq 0 \\ h & ,\text{if } \hat{t} = 0 \end{cases} \tag{3.2}$$

The input and output of each model is different from others. Also structure, number of parameters, and details of each model could be different from others. This flexibility in defining and customizing the models based on the time step during the prediction horizon (h) is one of the strength of our framework. For $i = 1, 2, \cdots, h$, we have:

- Model: M_i
- Input: $\{s^{t-\ell}, s^{t-\ell+1}, ..., s^{t-i}, \hat{s}^{t-i+1}, ..., \hat{s}^{t-1}\}$

  ($\ell - i + 1$ real values, $i - 1$ forecasted values)
- Algorithm: RNN with LSTM blocks
- Output: Forecast of output of all nodes, $\hat{s}^t$

$\hat{s}^t$ is output of the model M_i, which i is calculated from formula (3.2). If $t - l$ is greater than $t - i$ then we only use the last l predicted values. Also if $t - i + 1$ is equal to t, only real values are used. Loss function for each model M_i is defined as $L(M_i(\{s^{t-\ell}, ...\ s^{t-i}, \hat{s}^{t-i+1}, ...\ \hat{s}^{t-1}\}), x^t)$ where x^t is the real output of the nodes in the graph at time t and $L(a, b)$ is the mean absolute error of a and b. In figure 3.2 we show the overview of the model's detail.

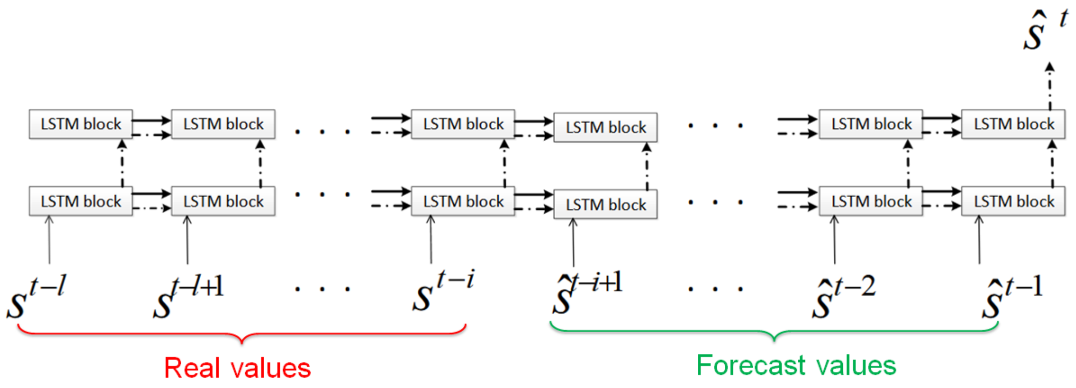

Figure 3.2: The model M_i trained at time t. Inputs are $s^{t-\ell}$, $s^{t-\ell+1}$, ..., s^{t-i} (real values) and $\hat{s}^{t-i+1}$, ... $\hat{s}^{t-1}$ (forecasted values from previous trained models). Black thick and dashed arrows are c_t and h_t based on Figure 3.1.

3.5 Experimemts and Results

We apply DL-STF to real wind speed data. East coast states are good candidates for our study as: (i) wind speed profiles are higher and (ii) there are more stations in a close vicinity in these states.

3.5.1 Data Description

We use hourly wind speed data from Meteorological Terminal Aviation Routine (METAR) weather reports of 57 stations in east coast including Massachusetts, Connecticut, New York, and New Hampshire [44]. Fig. 3.3 depicts the area under study and the location of these 57 stations. The target station Nantucket Memorial Airport (ACK) (circled in red) is located on an island and is subject to wind profiles with high ramps and speeds due to the fact that the surrounding surface has very low roughness heights. Furthermore, this area has good correlations with other stations owing to the fact the prevailing wind direction of this region is mainly northwest or southeast. A time period from January 6, 2014 to February 20, 2014 is considered as test set in our simulations. This time period has the most unsteady wind conditions throughout the year.

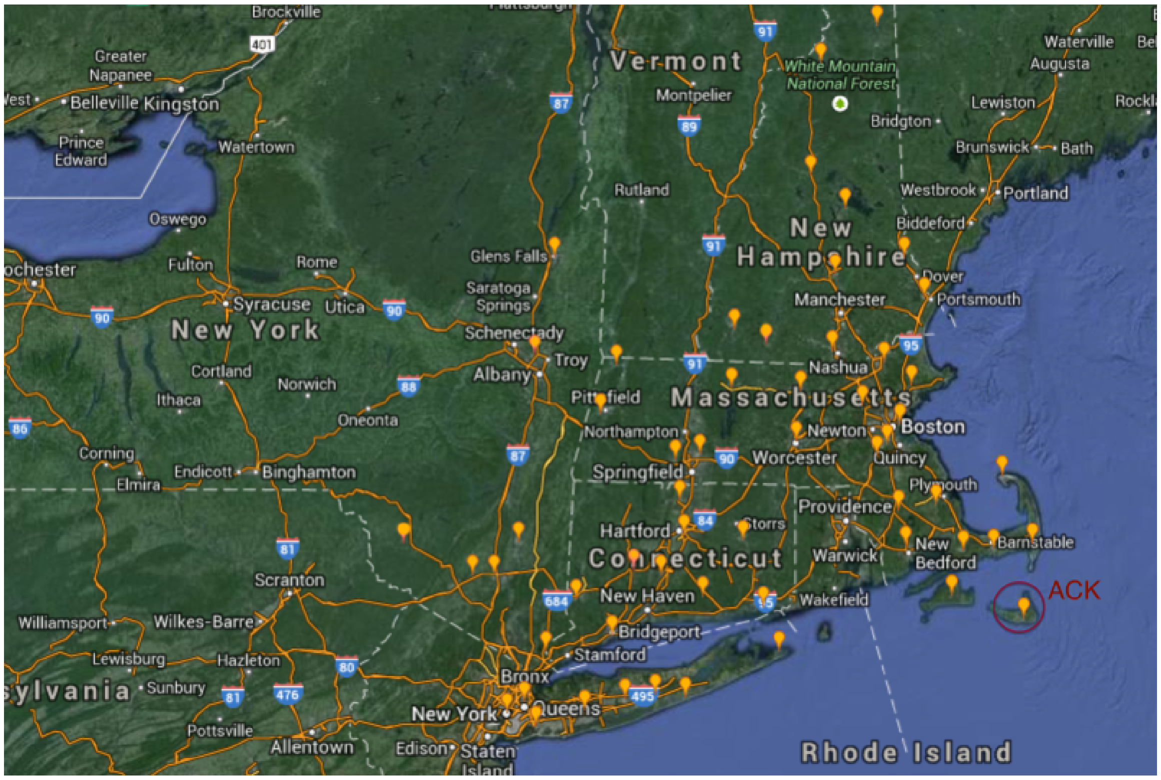
Figure 3.3: Map of the area under study. The 57 measuring locations in east coast are shown with yellow points. Circled in red is the target station ACK.

3.5.2 Results

In this section we discuss the details about our implementation and hyper-parameters setting. In our experiments $h = 6$ and we chose $\ell = 12$ based on a cross validation study. The optimizer is MSRProp which shows good performance for RNN with learning rate of 0.001. The activation function is ReLu. Data is normalized between 0 and 1. We use one fully connected layer on top of h_t features to create the desired output layer.We use TensorFlow [45] and Keras [46]. For models whose input includes predicted values, we need to increase the model capacity to help overcome over-fitting. Thus, we increase the number of neurons and layers and use stacked LSTM. To show performance of our algorithm we use three common error measure MAE, RMSE, and NRMSE. Table 3.1 shows comparison of three common error measures between proposed method and other methods. It is worth mentioning that other existing methods are trained to forecast only one node at a time but our method can forecast the output of all nodes in the graph at one time. More importantly, as we can see the error comparisons, our method has smaller error values compared to all other methods and outperform state-of-the-art results. More details about how other methods work are available in [47], [48].

Table 3.1: Error measures for different methods for one node (ACK)

Method	MAE (m/s)	RMSE (m/s)	NRMSE (%)
Persistence Forecasting	2.14	2.83	16.86
AR of order 1	2.07	2.76	16.44
AR of order 3	2.07	2.76	16.40
WT-ANN	1.82	2.47	14.68
AN-based ST	1.80	2.30	13.69
LS-based ST	1.72	2.20	13.08
DL-STF	1.63	2.19	**13.08**
DL-STF(All nodes)	**1.18**	**1.62**	16.28

Table 3.2 shows the average of three error measures for all nodes in the graph. To the best of our knowledge there is no other method capable of forecasting outputs of all nodes in a graph in one framework. The average of the error measures of all nodes is even better than error measures for one node(ACK) with relative improvement about 27% for MAE and RMSE.

Table 3.2: Average error measures over all locations using DL-STF

	MAE(m/s)	RMSE(m/s)	NRMSE(%)
Our method	1.18	1.62	16.28

In DL-STF, we model all information and the hidden interactions between nodes of the graph. As explained earlier, in a spatio-temporal setting we use information of all nodes to forecast one node's output in order to improve the forecasting performance as compared to the case when we only use one node's data (temporal setting). Table 3.3 illustrates a comparison between these two cases: 1) DL-STF trained on all nodes of the graph to forecast one node's output (node ACK) and 2) DL-STF trained with data only from node ACK and, thus, we don't count for hidden relationships between nodes. Table 3.3 shows that the error measures in case 1 (spatio-temporal forecasting) has significantly improved.

Table 3.3: Comparison of forecasting error measures. **First row**: train only on data from the node ACK and test on the node ACK. **Second row**: train on data from all nodes of the graph and test on the node ACK. **Third row**: train on data from all nodes and test on all nodes and calculate the average error measures over all nodes.

	MAE(m/s)	RMSE(m/s)	NRMSE(%)
one node	1.99	2.60	15.46
all nodes (ACK)	1.63	2.19	**13.08**
mean all nodes	**1.18**	**1.62**	16.28

In Figure 3.4 we show the accuracy of forecasting for 16 nodes of the graph. It shows real values vs forecast values on test data. The average of error measures are 1.203, 1.663,

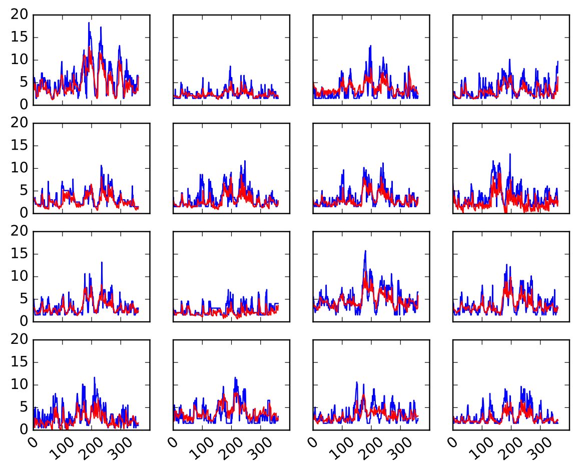
Figure 3.4: Forecasting performance on 16 stations. Blue lines are real values and red ones are forecast values. Horizontal axis shows time steps and vertical axis shows wind speed(m/s).

16.378 for MAE, RMSE, NRMSE respectively.

3.6 Conclusion and future works

So far, we model the spatio-temporal information by a graph whose nodes are data generating entities and its edges basically model how these nodes are interacting with each other. One of the main contributions of our work is the fact that we obtain forecasts of all nodes of the graph at the same time. Results of a case study on recorded time series data from a collection of wind mills in the north-east of the U.S. show that the proposed DL-based forecasting algorithm significantly improves the short-term forecasts compared to a set of widely-used benchmarks models.

There are two important differences between our proposed method compared to other methods: 1) existing methods forecast output of one node while our approach yields in forecasts of all nodes and, 2) most of the existing methods update during the input horizon and use the new data but our model does not need to update during input horizon which can improve the speed and performance of the algorithm.

For future we plan to extend our algorithm

- trying different architecture on the data
- trying to do more experiments with the other kinds of RNN like GRU
- trying to apply this model on traffic data

3.7 References

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recog- nition, 2009. CVPR 2009. IEEE Conference on, pp. 248– 255. IEEE, 2009.

Dowell, Jethro, Weiss, Stephan, Hill, David, and Infield, David. Short-term spatio-temporal prediction of wind speed and direction. Wind Energy, 2013.

Gneiting, Tilmann, Larson, Kristin, Westrick, Kenneth, Genton, Marc G., and Aldrich, Eric. Calibrated prob- abilistic forecasting at the stateline wind energy center: The regime-switching space–time method. Journal of the American Statistical Association, 101(475):968–979, 2006.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. Deep learning, 2016.

Graves, Alex. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

Grover, Aditya, Kapoor, Ashish, and Horvitz, Eric. A deep hybrid model for weather forecasting. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 379–386. ACM, 2015.

Hao, He, Sanandaji, Borhan M., Poolla, Kameshwar, and Vincent, Tyrone L. Aggregate flexibility of thermostat- ically controlled loads. IEEE Transactions on Power Systems, 2013.

He, Miao, Yang, Lei, Zhang, Junshan, and Vittal, Vijay. A spatio-temporal analysis approach for short-term forecast of wind farm generation. IEEE Transactions on Power Systems, 29(4):1611–1622, 2014.

Hering, Amanda S and Genton, Marc G. Powering up with space-time wind forecasting. Journal of the American Statistical Association, 105(489):92–104, 2010.

Hochreiter, Sepp and Schmidhuber, Jrgen. Long short term memory. Neural computation, 9(8):17351780, 1997. IEA, (International Energy Agency).

Iowa Environmental Mesonet. ASOS historical data, 2014. URL http://mesonet.agron.iastate.edu/ASOS/.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pp. 1097–1105, 2012.

Li, Gong and Shi, Jing. On comparing three artificial neural networks for wind speed forecasting. Applied Energy, 87 (7):2313–2320, 2010.

Ma, Xuejiao, Jin, Yu, and Dong, Qingli. A generalized dy- namic fuzzy neural network based on singular spectrum analysis optimized by brain storm optimization for short- term wind speed forecasting. Applied Soft Computing, 54:296–312, 2017.

Ren, Shaoqing, He, Kaiming, Girshick, Ross, and Sun, Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. In Conference on Neural Information Processing Systems (NIPS 2015), 2015. Sanandaji, Borhan M., Hao, He, Poolla, Kameshwar, and Vincent, Tyrone L. Improved battery models of an aggre- gation of thermostatically controlled loads for frequency regulation. In American Control Conference (ACC 2014), 2014.

Sanandaji, Borhan M, Tascikaraoglu, Akin, Poolla, Kamesh- war, and Varaiya, Pravin. Low-dimensional models in spatio-temporal wind speed forecasting. In American Control Conference (ACC 2015), pages 4485–4490. arXiv preprint arXiv:1503.01210, 2015.

Smith, J. Charles, Milligan, Michael R., DeMeo, Edgar A., and Parsons, Brian. Utility wind integration and operating impact state of the art. IEEE Transactions on Power Systems, 22(3):900–908, 2007.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pp. 3104–3112, 2014.

Tascikaraoglu, A and Uzunoglu, M. A review of combined approaches for prediction of short-term wind speed and power. Renewable and Sustainable Energy Reviews, 34: 243–254, 2014.

Tascikaraoglu, Akin, Sanandaji, Borhan M, Poolla, Kamesh- war, and Varaiya, Pravin. Exploiting sparsity of intercon- nections in spatio-temporal wind speed forecasting using wavelet transform. Applied Energy, 165:735–747, 2016. Tastu, Julija, Pinson, Pierre, Kotwa, Ewelina, Madsen, Hen- rik, and Nielsen, Henrik Aa. Spatio-temporal analysis and modeling of short-term wind power forecast errors. Wind Energy, 14(1):43–60, 2011.

Tastu, Julija, Pinson, Pierre, Trombe, P-J, and Madsen, Hen- rik. Probabilistic forecasts of wind power generation ac- counting for geographically dispersed information. IEEE Transactions on Smart Grid, 5(1):480–489, 2014.

Wang, HZ, Wang, GB, Li, GQ, Peng, JC, and Liu, YT. Deep belief network based deterministic and probabilistic wind speed forecasting approach. Applied Energy, 182:80–93, 2016.

Wytock, Matt and Kolter, Zico. Large-scale probabilistic forecasting in energy systems using sparse gaussian con- ditional random fields. Proceedings of the 52nd IEEE Conference on Decision and Control, pp. 1019–1024, 2013.

Xie, Le, Gu, Y., Zhu, X., and Genton, M. G. Short-term spatio-temporal wind power forecast in robust look-ahead power system dispatch. IEEE Transactions on Smart Grid, 5(1):511–520, 2014.

Zhang, Yao, Wang, Jianxue, and Wang, Xifan. Review on probabilistic forecasting of wind power generation. Renewable and Sustainable Energy Reviews, 32:255–270, 2014.

Zhu, Xinxin and Genton, Marc G. Short-term wind speed forecasting for power system operations. International Statistical Review, 80(1):2–23, 2012.

CHAPTER 4

Improving the Accuracy of the CogniLearn System for Cognitive Behavior Assessment

4.1 Introduction and problem definition

HTKS [6] is a game-like cognitive assessment method, designed for children between four and eight years of age. During the HTKS assessment, a child responds to a sequence of requests, such as “touch your head” or “touch your toes”. The cognitive challenge stems from the fact that the children are instructed to interpret these requests not literally, but by touching a different body part than the one stated. We propose some specific improvements that make the motion analysis more accurate. As a result of these improvements, the accuracy in recognizing cases where subjects touch their toes has gone from 76.46% in our previous work to 97.19%. A *Microsoft Kinect V2* camera is used for recording human motion. Then, we use the DeeperCut method [7] to perform body pose estimation in each frame. Finally, using the body pose estimates from DeeperCut we use a classification module that determines whether the subject touched his or her head, shoulders, knees, or toes. The system compares the part that was touched with the part that should have been touched based on the rules of the game, and assesses the overall accuracy score of the person playing the game.

4.2 Related Works

Several deep-learning methods have been proposed in recent years for video analysis and activity recognition [49, 50, 51], offering significantly improved accuracy compared to previous approaches[52, 53]. Deep learning methods have also been used in supervised or

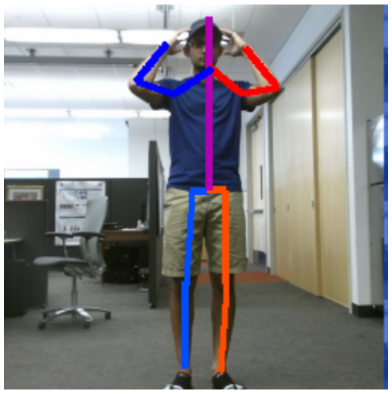

Figure 4.1: A sample human body pose estimation on a frame using DeeperCut [7]

unsupervised manner in different tasks in computer vision [7, 54], oftentimes producing state-of-the-art results.

In [8] we have introduced the CogniLearn system, which is used for automated video capture and performance assessment during the HTKS assessment. CogniLearn is designed to provide meaningful data and measures that can benefit therapists and cognitive experts. More specifically, the motion analysis and evaluation module provides systematic feedback regarding the performance of the HTKS tasks to the human experts. We build upon the CogniLearn system, and we suggest some specific improvements in the motion analysis module, that lead to higher recognition accuracy.

4.3 Our Method

We use DeeperCut [7] to estimate the location of human body parts in each color frame of the video. Figure 4.1 shows a video frame where we have superimposed the body part locations estimated by DeeperCut. Each color frame of a test video sequence is provided as input to the DeeperCut method. The output of the algorithm is the image location of 12 body parts: head, shoulder(right and left), elbow(right and left), wrist(right and left), hip, knee(right and left), ankle(right and left).

After we obtain the body part locations from DeeperCut, we perform an additional step, in order to estimate whether the human, at that frame, is touching his or her head, shoulders, knees, or toes. As a first step, we define a distance D between hands and head, hands and shoulder, hands and knees, and hands and ankles. Using $||\cdot||$ to denote Euclidean norms, this distance is defined as follows:

$$D(\text{head}) = \frac{||\text{lh} - \text{head}|| + ||\text{rh} - \text{head}||}{2} \tag{4.1}$$

$$D(\text{shoulders}) = \frac{||\text{lh} - \text{ls}|| + ||\text{rh} - \text{rs}||}{2} \tag{4.2}$$

$$D(\text{knees}) = \frac{||\text{lh} - \text{lk}|| + ||\text{rh} - \text{rk}||}{2} \tag{4.3}$$

$$D(\text{ankles}) = \frac{||\text{lh} - \text{la}|| + ||\text{rh} - \text{ra}||}{2} \tag{4.4}$$

In the above definitions, head stands for the (x, y) pixel location of the center of the head in the color frame, as estimated by DeeperCut. Similarly, lh and rh stand for the locations the left and right hand, ls and rs stand for the locations of the left and right shoulder, lk and rk stand for the locations of the left and right knee, and la and ra stand for the locations of the left and right ankle.For example, $||\text{lh} - \text{head}||$ denotes the Euclidean distance between the left hand and the center of the head.

Based on these D values, one approach for estimating the body part that is being touched is to simply select the body part for which the D score is the smallest. This was the approach used in [8]. However, when the person touches the toes or knees, this approach does not work well. When a person bends down to touch the knees or toes with the hands, the head inevitably also gets near to the knees or toes. In that case, two issues may arise. The first one is that the accuracy of the body joint estimator is decreased. The second issue is that the detected location for the head is near the detected locations for the knees or toes. As a result, for example, when the hands are touching the toes, it frequently happens that the distance of hands to the head is estimated to be smaller than distance of the hands to the toes. These two issues can lead to inaccuracies. As we see in Table 4.1, in the original CogniLearn results of [8], 9.33% of toe frames are classified as head frames, and 14.00% of toe frames are classified as knee frames.

We propose two rules to improve the classification accuracy of toe frames:

Rule 1: If the distance between the head and the hip is less than a predefined threshold, we can immediately conclude that the hands are touching the toes.

Rule 2: Sometimes, when the hands are touching the head, the distance between the hands and the head is estimated to be longer than the distance between the hand and the shoulders. To address this issue, we add a constant bias value to the distance between hands and shoulders, before comparing it with the distance between the hands and the head.

In the experiments, we demonstrate that these two rules significantly improve the classification accuracy on toe and head frames, while only minimally affecting the classification accuracy on frames where the hands touch the shoulders or knees.

4.4 Experiments

For our experiments, we use the same dataset that was used in the original CogniLearn paper [8]. The dataset includes color videos from 15 participants, whose ages are between

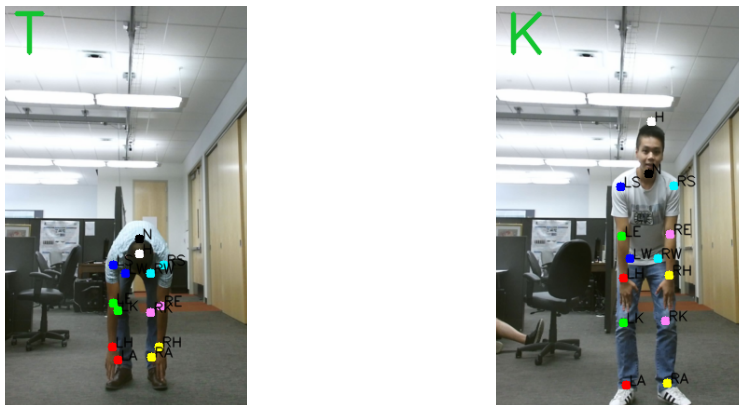

Figure 4.2: Results using the full method described, i.e., when both Rule 1 and Rule 2 are used. On the left, we see a frame where the hands touch the toes. On the right, we see a frame where the hands touch the knees. The green letter on the top left of each frame is the classification output of the system, where “T” stands for “toes”, “K” stands for “knees”.

18 and 30 years (while the HTKS assessment has been designed for children between the ages of 4 and 8, at this time we still do not have recorded data available from children of that age). In total, the dataset contains over 60,000 video frames. Figure 4.2 shows examples of test frames correctly recognized by our algorithm. The green letter in top left of the images shows the classification output of our system (“T” stands for “toes”, “K” stands for “knees”).

Our method is applied on each color frame separately. The goal of our method is to classify each frame into one of four classes, corresponding respectively to whether the human is touching his or her head, shoulders, knees, or toes. Ground truth information is provided for 4,443 video frames, and we use those frames as our test set. The ground truth specifies, for each frame, which of the four classes belongs to. Accuracy is simply measured as the percentage of test frames for which the output of our system matched the ground truth.

We should emphasize that the results that we present are user-independent. None of the 15 subjects appearing in the test set is used to train any part of our models. The only module that uses training is DeeperCut, and we use the pretrained model that has been made available by the authors of [7].

4.4.1 Results

Table 4.1 shows the confusion matrix reported in the paper [8]. As we can see in that table, shoulder and knee frames are recognized at rather high accuracies of 99.63% and 98.17% respectively. However, head and toes frames are recognized with lower accuracies, 94.47% and 76.46% respectively. Our project was primarily motivated by the need to improve the accuracy for those two cases.

Table 4.1: Confusion matrix reported by [8]. Rows correspond to ground truth labels, and columns correspond to classification outputs.

		Recognized				
		Head	Shoulder	Knee	Toe	Sum
Real	Head	**94.47**	5.53	0.00	0.00	100
	Shoulder	0.12	**99.63**	0.25	0.00	100
	Knee	0.00	0.54	**98.17**	1.29	100
	Toe	9.33	0.21	14.00	**76.46**	100

In Table 4.2 we report the results from the proposed method (i.e, when we apply both Rule 1 and Rule 2 from Section 4.3. As we can see, the accuracy for all four categories is more than 94.7%. The accuracy for head frames is marginally improved compared to [8]. The accuracy for shoulder and knee frames is slightly worse compared to [8]. At the same time, the accuracy for toe frames is now 97.19%, significantly higher than the accuracy of 76.46% reported in [8]. Finally, in Table 4.3 we show results using a partial implementation of our method, applying only Rule 1, and not Rule 2. We note that the overall accuracy is mostly similar to what we get when we combine Rules 1 and 2. Overall,

Table 4.2: Confusion matrix obtained using the full method described, i.e., when both Rule 1 and Rule 2 are added to the method of [8]. Rows correspond to ground truth labels, and columns correspond to classification outputs.

		Recognized				
		Head	Shoulder	Knee	Toe	Sum
Real	Head	**94.78**	3.39	0.26	1.57	100
	Shoulder	0.50	**99.25**	0.12	0.12	100
	Knee	0.00	0.60	**97.22**	2.18	100
	Toe	0.76	0.00	2.05	**97.19**	100

Table 4.3: Confusion matrix obtained by adding Rule 1 to the method of [8]. Rows correspond to ground truth labels, and columns correspond to classification outputs.

		Recognized				
		Head	Shoulder	Knee	Toe	Sum
Real	Head	**93.21**	4.96	0.26	1.57	100
	Shoulder	0.37	**99.39**	0.12	0.12	100
	Knee	0.00	0.60	**97.22**	2.18	100
	Toe	0.76	0.00	2.05	**97.19**	100

Rule 1 is by far the biggest contributor to the improvements we obtain over the original results of [8]. At the same time, the accuracy for head frames improves from 93.21% to 94.78% when we use Rules 1 and 2, compared to using only Rule 1. Rule 2 was explicitly designed to reduce the percentage of head frames that were classified as shoulder frames. Indeed, using Rule 2 (together with Rule 1) reduces that percentage from 4.96% (obtained using only Rule 1) to 3.39%. Table 4.4 shows the overall classification accuracy. In that table, the overall accuracy is defined as the average of the accuracies over the four different classes. The overall accuracy improves from the 92.18% rate of [8] to 96.75% when we add Rule 1, and to 97.11% when we also add Rule 2.

Table 4.4: Comparisons in accuracy between the original results of [8], the results obtained by adding Rule 1 to the method of [8], and the results obtained by adding both Rule 1 and Rule 2 to the method of [8]

	Overall	**H**	**S**	**K**	**T**
Original[8]	92.18	94.47	**99.63**	**98.17**	76.46
Rule 1	96.75	93.21	99.39	97.22	97.19
Rules 1,2 combined	**97.11**	**94.78**	99.25	97.22	**97.19**

Figure 4.3 shows some sample test frames. More specifically, from each of the four classes we show an example that was classified correctly, and an example that was classified incorrectly. We note that separating the head from the shoulder class can be quite challenging at times, because the distribution of hand positions does not vary much between the two classes. Separating knees and toes can also be difficult, because in frames belonging to both classes the knees are typically occluded, and there is significant overlap between the arms and the legs. This leads to errors in the estimated positions of the hands and the knees.

4.5 Conclusions and Future Works

We have propose a method for improving the accuracy of the original CogniLearn[8] system in recognizing, for each video frame, whether the human is touching the head, shoulders, knees, or toes in that frame. The experiments have shown that our improvements lead to significantly better accuracy, especially for frames where the human touches the toes. In those cases, the accuracy increased from the 76.46% rate in [8] to 97.19%.

Our project of automatically capturing and analyzing performance in the HTKS test is still in its initial stages.

1. A high priority for us is to obtain data from children between the ages of 4 and 8, as that is the target age group for the HTKS test.

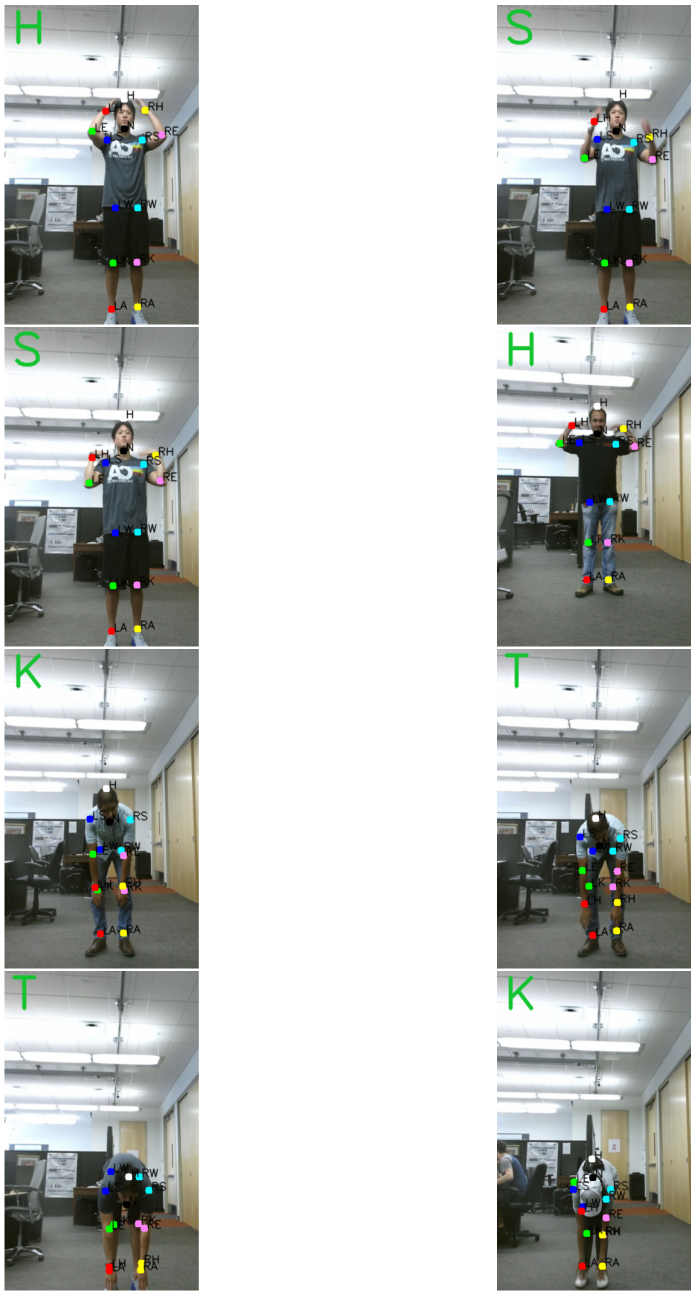

Figure 4.3: Example test frames, with the classification output superimposed. The classification output is correct for the examples on the left column, and incorrect for the examples on the right column. The ground truth is: “head” for row 1, “shoulders” for row 2, “knees” for row 3, “toes” for row 4

2. Also, we plan to explore using the depth modality of the Kinect camera in addition to the color modality that we have used so far.
3. Finally, we should note that the HTKS assessment includes a "self-correction" category, in which the subject has started doing an incorrect motion and then self-corrected [6]. In the near future we plan to work on developing methods for identifying such self-correction cases, so that our assessment fully matches the formal HTKS description.

4.6 References

[1] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2625–2634, 2015.

[2] M. Fouladgar, M. Parchami, R. Elmasri, and A. Ghaderi. Scalable deep tra c flow neural networks for urban tra c congestion prediction. In International Joint Conference on Neural Networks (IJCNN). IEEE, 2017.

[3] S. Gattupalli, D. Ebert, M. Papakostas, F. Makedon, and V. Athitsos. Cognilearn: A deep learning-based interface for cognitive behavior assessment. In intelligent user interfaces, 2017.

[4] S. Gattupalli, A. Ghaderi, and V. Athitsos. Evaluation of deep learning based pose estimation for sign language. In Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments. ACM, 2016.

[5] A. Ghaderi and V. Athitsos. Selective unsupervised feature learning with convolutional neural network (S-CNN). In International Conference on Pattern Recognition(ICPR), 2016.

[6] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. arXiv preprint arXiv:1605.03170, 2016.

[7] D. Leightley, J. Darby, B. Li, J. S. McPhee, and M. H. Yap. Human activity recognition for physical rehabilitation. In 2013 IEEE International Conference on Systems, Man, and Cybernetics, pages 261–266. IEEE, 2013.

[8] M. M. McClelland and C. E. Cameron. Self-regulation in early childhood: Improving conceptual clarity and developing ecologically valid measures. Child Development Perspectives, 6(2):136–142, 2012.

[9] M. M. McClelland, C. E. Cameron, R. Duncan, R. P. Bowles, A. C. Acock, A. Miao, and M. E. Pratt. Predictors of early growth in academic achievement: The head-toes-knees-shoulders task. Frontiers in psychology, 5, 2014.

[10] L. Xia and J. Aggarwal. Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2834–2841, 2013.

CHAPTER 5

Face Recognition at Scale

5.1 Introduction

Face recognition and verification is one of the popular tasks in computer vision. Many researchers are interested in face recognition. Also, this topic is very applicable in many applications like cameras in airports or city cameras for security purposes. So face recognition with huge number of classes are very interesting and applicable topic in computer vision. Our goal is to propose an end to end method for face recognition for about one million identities. This chapter introduces datasets for face recognition and face verification and review some recent papers using deep learning methods for face recognition. Then we report our results in different datasets and propose a plan for our future research path.

5.2 Related Works

This section summarize some recent papers for face recognition. Nearly all state of the art methods for face recognition nowadays use deep learning methods.

Recent work using Generative Adversarial Network (GAN) [55] for face recognition is found in [56] which names as DR-GAN. Main contribution of the work is using GAN to generate features for face recognition and generation. With one or multiple face images as the input, DR-GAN produce an identity representation that is discriminative and generative.The framework combines 1- learning pose invariant identity representation 2- synthesis faces with arbitrary poses. At test time despite of other methods, this method can take any number of images and create a representation. They reported the state of the art in Multi-PIE, CFP, and IJB-A datasets.

Liu [57] proposes a framework called SphereFace. They proposed a angular softmax as loss function to increase intra class and decrease inter class distances. For experiments, they use the Caffe and report results on LFW, YTF, MegaFace Challenge 1 datasets. The face landmarks in all images are detected by MTCNN. They train the network with angular softmax for extracting features. Then for testing, they use deep features with cosine similarity as angular metric. For data augmentation the images were horizontally flipped.

Yang [58] introduced an attention based method for face verification and identification on a video. Creating fixed size feature vector for video undependable of the length and suitable for recognition is one of the main contributions of the work. Despite of other methods, they do not fix the weights or use heuristic methods, they use neural network to adaptively calculate weights. It has 2 module 1) frame level feature extraction 2) feature aggregation and create video level feature. They generate a compact video face representation for a video.

Yin and Liu [59] used a multi-task CNN for face recognition where identity recognition is the main task and pose, illumination, and expression estimations are the side tasks.A dynamic-weighting scheme to automatically assign the loss weight to each side task was designed. The paper introduced a pose-directed multi-task CNN by grouping different poses to learn pose-specific identity features, simultaneously across all poses.

Multi task learning tries to train multi tasks simultaneously to improve performance of one or all of them. The main task here is face recognition and side tasks are Pose, Illumination, Expression(PIE). They suppose different tasks share the same features. The features before the softmax layer are used for face matching based on cosine similarity. The proposed method was evaluated on the public datasets: LFW, CFP, and IJB-A.

Schroff et al. in the paper [9] used 1*1*d convolution layers and inception model of Szegedy et al [60] which reduce the number of parameters by up to 20 times and end to end training of the network. They used triplet loss for verification, recognition and clustering of faces and report the results on the You Tube Faces, LFW, and a private dataset.

5.3 Face datasets

There is a close relationship between face recognition algorithms and the availability of datasets. There are many different datasets with different aspects. For example YouTube Faces Dataset (YFD) contains 3425 videos of 1595 different people which all videos were downloaded from the YouTube and for each subject there are 2.15 videos. CelebFaces dataset is consists of 202599 images of 10177 celebrities. Each image has 30 attribute annotations like eyeglasses, wearing hat, smiling, landmarks, etc. DeepFace, VGGFace, FaceNet datasets are private. LFW dataset includes 13233 face images from 5749 different identities. In table 5.1 we summarize name, availability, size, number of images, number of individuals, and number of videos of some popular face datasets. As the table 5.1 shows, the dataset MegaFace has the most number of individuals among the public datasets. The maximum number of identities before MegaFace was 100K, while MegaFace has 672K. Our goal is to propose a method for face recognition at scale. The MegaFace dataset has been recently released and it is very suitable for our goal.

5.4 Our method

For scale face recognition, we have thousand or millions of classes so it is not possible to have these number of classes in recognition layer as a softmax layer (like usual smaller classification networks). One of the ways to overcome this problem, is using the triplet loss (as a kind of metric learning) to map each face to a new space. This mapping tries to minimize

Table 5.1: List of some popular face datasets

Name	public	size	#of images	#of individuals	# of videos	celebrity
LFW	Y	8.6GB	13233	5749	-	N
IJB-B	Y	106GB	5396	500	2042	N
Multi PIE	Y	305GB	754200	337	-	N
CASIA-Webface	Y		494414	10575	-	N
Mega Face	Y	900 GB	4.7 M	672K	-	N
YouTube (YFD)	Y		-	1595	3425	N
YouTube Celebrities	Y		-	47	1910	Y
MS-Celeb-1M	Y	200GB	10M	100K	-	N
CASIA NIR-VIS 2.0	Y		17580 pairs	725	-	N
CFP	Y		7000 pairs	500	-	Y
COX	Y		1000	4000	-	N
FaceScrub	Y		100K	530	-	Y
CelebFaces	Y		202K	10K	-	Y
UMD Faces	Y	260GB	367K	8.2K/3K	22K	Y
VGG-Face	Y		2.6M	2.6K	-	Y
SFC(FB)	N		4.4M	4K	-	N
Google	N		200M	8M	-	N
Adience	N		26K	2.2K	-	N

the (Euclidean) distance of similar faces and maximize distance of different faces. So there is no recognition layer. There is a feature layer before triplet loss layer. The dimension of embedding layer (e.g. 128, ... 1024 or 4096) is less than input size. 128-dimensional-bytes vector needs less computational cost and improve performance. After training the network for generating embedding features, cosine similarity or Euclidean distance between embedding of faces with a threshold is used to recognition and verification:

1. Recognition : find the nearest neighbor with K-NN algorithm
2. Verification : using a threshold for distance of two faces

5.5 Experiments

We have extensive experiments with different number of classes. For feature extraction we use the model trained on CASIA dataset. The number of embedding for each image is 128. So each image converted to a 128-dimensional vector. We have trained SVM, KNN-standard, and KNN-central for classification. Also we tried different datasets to report the performance of the algorithm. We have a range of experiments with the 500 number of classes to 200K number of classes. For the first experiment we use the embedding from trained network on CASIA dataset and then train a SVM classifier for classification on Face Scrub dataset.

5.5.1 Experiments on FaceScrub

Table 5.3 shows the results of the algorithm on the 526 classes of FaceScrub dataset. This dataset has 530 classes. We select the classes which contain at least 40 samples and consider 35 samples for training and the rest for testing. KNN and SVM show similar performance in the term of accuracy but in the term of the time KNN is outperforming the SVM.

5.5.2 Experiments on LFW

The LFW dataset contains 13233 images of 5749 individuals. The average number samples per individual is 2. So we have to decrease number of samples for training and testing. The classes with at least 5 samples were selected which 3 samples are for training and others are for testing.We have tried two version of KNN classifier. One is standard version which is showed as "KNN-S". The second version of KNN is using the center of

layer	**size-in**	**size-out**	**kernel**
conv1	220×220×3	110×110×64	7×7×3, 2
pool1	110×110×64	55×55×64	3×3×64, 2
conv2a	55×55×64	55×55×64	1×1×64, 1
conv2	55×55×64	55×55×192	3×3×64, 1
norm2	55×55×192	55×55×192	
pool2	55×55×192	28×28×192	3×3×192, 2
conv3a	28×28×192	28×28×192	1×1×192, 1
conv3	28×28×192	28×28×384	3×3×192, 1
pool3	28×28×384	14×14×384	3×3×384, 2
conv4a	14×14×384	14×14×384	1×1×384, 1
conv4	14×14×384	14×14×256	3×3×384, 1
conv5a	14×14×256	14×14×256	1×1×256, 1
conv5	14×14×256	14×14×256	3×3×256, 1
conv6a	14×14×256	14×14×256	1×1×256, 1
conv6	14×14×256	14×14×256	3×3×256, 1
pool4	14×14×256	7×7×256	3×3×256, 2
concat	7×7×256	7×7×256	
fc1	7×7×256	1×32×128	maxout p=2
fc2	1×32×128	1×32×128	maxout p=2
fc7128	1×32×128	1×1×128	
L2	1×1×128	1×1×128	

Table 5.2: The structure of the model for feature training based on [9] the stride and pooling size are $p = 2$.

Table 5.3: The results of experiments on the Face Scrub dataset

	SVM	KNN-C
Min #images in class	40	40
# image for train	35	35
#classes	526	526
#images train	18410	18410
#images test	73302	73302
Train time(min)	23	25
Test time(min)	121	103
Accuracy(%)	97.5	97.6

the training samples as one point and for testing we calculate the distance between query sample and center points.The class which has minimum distance is the classification result. The table 5.4 shows the results for SVM, KNN-C, and KNN-S.For the 423 classes of the LFW dataset, SVM shows the %97.5 accuracy. In term of time, the KNN-C shows better performance.

Table 5.4: The results of experiments on the 423 classes of the LFW dataset

	SVM	KNN-C	KNN-S
Min #images in class	5	5	5
# image for train	3	3	3
#classes	423	423	423
#images train	1269	1269	1269
#images test	4716	4716	4716
Train time(m:s)	2:20	1:30	1:26
Test time(m:s)	7:10	5:10	5:14
Accuracy(%)	97.5	99.5	99.3

5.5.3 Experiments on Mega Face

The Mega Face dataset is one of the largest public face datasets. Our goal is to improve the state of the art results in this dataset. For making accurate and reliable results, we limit the number of samples per class for training process to 3 and 2 and the number of samples per class for testing process to 2 and 1. The table 5.5 shows the results. When we investigate the results for MegaFace dataset, we found the there are some duplicate images in the dataset. The number of duplicate images are 42842.

Table 5.5: The results of experiments on the Mega Face dataset

	KNN-C	SVM	KNN-C	KNN-C	KNN-C	KNN-C
#images/class(train)	3	3	3	3	2	2
#images/class(test)	2	2	2	2	1	1
#classes	5K	5K	10K	20K	100k	663386
#images train	15K	15K	30K	60K	200k	1326772
#images test	10K	10K	20K	40K	100k	10K
Train time(m)	14	25	24	49	168	15h
Test time(m)	9	238	16	36	2193	9
accuracy(%)	75	45.0	72.6	69.9	62.2	54.6

5.6 Future works

We have done experiments on different number of classes on different datasets like FLW, Mega Face, and Face Scrub. The number of classes were 500, 5K, 10K, 20K, 100K, and 663386. Our goal is to build an end to end system for face recognition at scale. There are many algorithms which are suitable for face recognition with limited amount of classes e.g. 500 or 1000. One of the problems that we have in this project is that we want to find the performance of a method in much more number of classes. We think this is an interesting problem and we did not see some solution for it in literature. So it will be one of our partial goal in this project. Our Plan for future works

1. Adding distractors of the Mega Face dataset to the test set
2. Improving the accuracy of the face recognition by different structure for features extraction
3. Creating a method for proposing many classes with their probabilities for each test image instead of one class
4. Finding a way to predict the performance of a method on N classes based on performance on N/5 classes

CHAPTER 6

Conclusion

This thesis investigates improvements in application of deep learning in many areas such as unsupervised learning, forecasting, and face recognition. Our contribution so far are as follow:

- Proposing a new method for unsupervised feature learning
- Running extensive amount of experiments with caffe framework with different hyper parameters and apply transfer learning to trained models for showing power of our method for unsupervised feature learning
- Designing and implementing a new framework for forecasting to predict output values for all nodes in a graph
- Improvement in vision module of CogniLearn system for Cognitive Behavior Assessment
- Proposing a framework for face recognition for more than 650K individuals

6.1 Future works

The second focus of this work is on goals:

1. Unsupervised Feature Learning: (1 month)
   (a) Training our model on data from the ImageNet dataset
   (b) Running the experiments with bigger network structures
2. Forecasting:
   (a) Trying to report the results of our model on different datasets

(b) Testing the model in different problems which can modeled as a graph like traffic prediction

3. Cognitive Behavior Assessment:
   (a) collecting data from children between the ages of 4-8.
   (b) Using the depth modality in addition to the color modality
   (c) Identifying "Self-correction" cases, where the subject starts moving hands towards the wrong part, but then self-corrects the motion.
4. Face Recognition at Scale:
   (a) Adding distractors of the Mega Face dataset to the test set
   (b) Improving the accuracy of the face recognition by different structure for features extraction
   (c) Creating a method for proposing many classes with their probabilities for each test image instead of one class
   (d) Finding a way to predict the performance of a method on N classes based on performance on N/5 classes

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[4] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*, 2014, pp. 647–655.

[6] M. M. McClelland, C. E. Cameron, R. Duncan, R. P. Bowles, A. C. Acock, A. Miao, and M. E. Pratt, "Predictors of early growth in academic achievement: The head-toes-knees-shoulders task," *Frontiers in psychology*, vol. 5, 2014.

[7] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," *arXiv preprint arXiv:1605.03170*, 2016.

[8] S. Gattupalli, D. Ebert, M. Papakostas, F. Makedon, and V. Athitsos, “Cognilearn: A deep learning-based interface for cognitive behavior assessment,” in *intelligent user interfaces*, 2017.

[9] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[10] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[11] W. Zou, S. Zhu, K. Yu, and A. Y. Ng, “Deep learning of invariant features via simulated fixations in video,” in *Advances in neural information processing systems*, 2012, pp. 3203–3211.

[12] L. Bo, X. Ren, and D. Fox, “Unsupervised feature learning for rgb-d based object recognition,” in *Experimental Robotics*. Springer, 2013, pp. 387–402.

[13] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning.”

[14] K. Sohn and H. Lee, “Learning invariant representations with local transformations,” *arXiv preprint arXiv:1206.6418*, 2012.

[15] Q. V. Le, “Building high-level features using large scale unsupervised learning,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8595–8598.

[16] B. Alexe, T. Deselaers, and V. Ferrari, “Measuring the objectness of image windows,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.

[17] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[18] I. Endres and D. Hoiem, “Category independent object proposals,” *Computer Vision–ECCV 2010*, pp. 575–588, 2010.

[19] A. Krizhevsky, “Learning multiple layers of features from tiny images,” 2009.

[20] Y. Jia, C. Huang, and T. Darrell, “Beyond spatial pyramids: Receptive field learning for pooled image features,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3370–3377.

[21] K. Swersky, J. Snoek, and R. P. Adams, “Multi-task bayesian optimization,” in *Advances in neural information processing systems*, 2013, pp. 2004–2012.

[22] A. Coates and A. Y. Ng, “Selecting receptive fields in deep networks,” in *Advances in Neural Information Processing Systems*, 2011, pp. 2528–2536.

[23] X. Zhu and M. G. Genton, “Short-term wind speed forecasting for power system operations,” *International Statistical Review*, vol. 80, no. 1, pp. 2–23, 2012.

[24] A. Tascikaraoglu and M. Uzunoglu, “A review of combined approaches for prediction of short-term wind speed and power,” *Renewable and Sustainable Energy Reviews*, vol. 34, pp. 243–254, 2014.

[25] T. Gneiting, K. Larson, K. Westrick, M. G. Genton, and E. Aldrich, “Calibrated probabilistic forecasting at the stateline wind energy center: The regime-switching space–time method,” *Journal of the American Statistical Association*, vol. 101, no. 475, pp. 968–979, 2006.

[26] A. S. Hering and M. G. Genton, “Powering up with space-time wind forecasting,” *Journal of the American Statistical Association*, vol. 105, no. 489, pp. 92–104, 2010.

[27] L. Xie, Y. Gu, X. Zhu, and M. G. Genton, "Short-term spatio-temporal wind power forecast in robust look-ahead power system dispatch," *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 511–520, 2014.

[28] J. Dowell, S. Weiss, D. Hill, and D. Infield, "Short-term spatio-temporal prediction of wind speed and direction," *Wind Energy*, 2013.

[29] M. He, L. Yang, J. Zhang, and V. Vittal, "A spatio-temporal analysis approach for short-term forecast of wind farm generation," *IEEE Transactions on Power Systems*, vol. 29, no. 4, pp. 1611–1622, 2014.

[30] J. Tastu, P. Pinson, E. Kotwa, H. Madsen, and H. A. Nielsen, "Spatio-temporal analysis and modeling of short-term wind power forecast errors," *Wind Energy*, vol. 14, no. 1, pp. 43–60, 2011.

[31] J. Tastu, P. Pinson, P.-J. Trombe, and H. Madsen, "Probabilistic forecasts of wind power generation accounting for geographically dispersed information," *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 480–489, 2014.

[32] M. Wytock and Z. Kolter, "Large-scale probabilistic forecasting in energy systems using sparse gaussian conditional random fields," *Proceedings of the* 52*nd IEEE Conference on Decision and Control*, pp. 1019–1024, 2013.

[33] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[34] H. Wang, G. Wang, G. Li, J. Peng, and Y. Liu, "Deep belief network based deterministic and probabilistic wind speed forecasting approach," *Applied Energy*, vol. 182, pp. 80–93, 2016.

[35] A. Grover, A. Kapoor, and E. Horvitz, "A deep hybrid model for weather forecasting," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 379–386.

[36] X. Ma, Y. Jin, and Q. Dong, “A generalized dynamic fuzzy neural network based on singular spectrum analysis optimized by brain storm optimization for short-term wind speed forecasting,” *Applied Soft Computing*, vol. 54, pp. 296–312, 2017.

[37] G. Li and J. Shi, “On comparing three artificial neural networks for wind speed forecasting,” *Applied Energy*, vol. 87, no. 7, pp. 2313–2320, 2010.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[39] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Conference on Neural Information Processing Systems (NIPS 2015)*, 2015.

[40] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 248–255.

[42] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[43] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” 2016.

[44] Iowa Environmental Mesonet, “ASOS historical data,” 2014. [Online]. Available: http://mesonet.agron.iastate.edu/ASOS/

[45] M. Abadi and et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[46] F. Chollet *et al.*, “Keras,” https://github.com/fchollet/keras, 2015.

[47] B. M. Sanandaji, A. Tascikaraoglu, K. Poolla, and P. Varaiya, "Low-dimensional models in spatio-temporal wind speed forecasting," in *American Control Conference (ACC 2015), pages 4485–4490.* arXiv preprint arXiv:1503.01210, 2015.

[48] A. Tascikaraoglu, B. M. Sanandaji, K. Poolla, and P. Varaiya, "Exploiting sparsity of interconnections in spatio-temporal wind speed forecasting using wavelet transform," *Applied Energy*, vol. 165, pp. 735–747, 2016.

[49] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2625–2634.

[50] S. Gattupalli, A. Ghaderi, and V. Athitsos, "Evaluation of deep learning based pose estimation for sign language," in *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments.* ACM, 2016.

[51] M. Fouladgar, M. Parchami, R. Elmasri, and A. Ghaderi, "Scalable deep traffic flow neural networks for urban traffic congestion prediction," in *International Joint Conference on Neural Networks (IJCNN).* IEEE, 2017.

[52] D. Leightley, J. Darby, B. Li, J. S. McPhee, and M. H. Yap, "Human activity recognition for physical rehabilitation," in *2013 IEEE International Conference on Systems, Man, and Cybernetics.* IEEE, 2013, pp. 261–266.

[53] L. Xia and J. Aggarwal, "Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2834–2841.

[54] A. Ghaderi and V. Athitsos, "Selective unsupervised feature learning with convolutional neural network (S-CNN)," in *International Conference on Pattern Recognition(ICPR)*, 2016.

[55] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[56] L. Tran, X. Yin, and X. Liu, “Disentangled representation learning gan for pose-invariant face recognition.”

[57] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “Sphereface: Deep hypersphere embedding for face recognition,” *CVPR*, 2017.

[58] J. Yang, P. Ren, D. Chen, F. Wen, H. Li, and G. Hua, “Neural aggregation network for video face recognition,” *CVPR*, 2017.

[59] X. Yin and X. Liu, “Multi-task convolutional neural network for face recognition,” *CVPR*, 2017.

[60] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.