T-10

Facial expression recognition using CNN

Anirudh Kothapalli (20093112), Marulasidda Swamy K S (20116267), Naveenkumar A R (20098468)

IFT 6390 – Fondements de l'apprentissage machine (Automne 2018)

Professor: Mr. Ioannis Mitliagkas

Université de Montréal

Abstract

Convolution Neural Network is trying to solve many real time applications. Facial expressions recognition is one of the applications where in researchers can achieve 80% accuracy in prediction. We would like to work on this project to know more about working of CNN, preprocessing techniques involved in understanding data in high dimensional space, classifiers involved in facial recognition etc. As part of the project, we will be working on some new concepts like adaptive gradient, cascade classifier for facial recognition.

Introduction

The goal of the project was to implement a classification algorithm that can automatically predict the categories of human expressions given human facial images. The CK+ dataset [5] consists of total 648 images and each image is labelled with one of the eight basic expressions("neutral", "anger", "contempt", "disgust", "fear", "happy", "sadness", "surprise"). As the first step we have prepared the data from CK+ data set which can be directly fed to any machine learning algorithm.

We have done the data preparation in three steps:

1) Copy images to respective folder as per their labelled expression.
2) Detect human face in each image using cascade classifier and cut the image to the size of human face detected.
3) Divide the data in to train and test data of size 521 and 127 respectively. Each image is of size 350px x 350px and these images have been modified by converting in to grayscale images.

We tried with the same CNN architecture with which we got good results in Kaggle competition(This is called Transfer learning), and we were able to get approximately 60% test accuracy. The CNN model with current architecture performed better than any other architectures with the test accuracy of 88.976% as test accuracy.

Feature Design

•Each item of the input data was reshaped to 48 x 48 arrays

Reason : Spatial relationship between the pixel values is not disturbed after resizing. Convolution is expensive with high dimension data.

•Conversion to Python Imaging Library (PIL / Pillow) format

Reason : To transform the data to be interpreted as images for the CNN training

Data augmentation was done with following operations :

•Random horizontal flipping

•Random vertical flipping

Reason : To enrich the dataset with possible variants of the input images in order to allow the CNN filters to be robust enough to learn and predict the labels even if the images are slightly rotated or flipped.

CNN models are translation invariant but are prone to variation in rotation and Scaling. So the networks need to

•Normalization of image pixel intensity values.

•The pixel intensity values of the images are maintained between 0 to 1.

Reason : To transform the input pixel values to a reasonable range to help the neural network to learn faster since gradients act uniformly.

As a CNN model is being used, the features are not hand-designed and the features with their hierarchies are learned by the filters of the convolutional layers of the CNN during the training process.

Algorithm

Given below are overviews of the learning algorithm used

Convolutional Neural Networks (CNNs)

We have taken the methods of horizontal and vertical flip to get the augmented data for application in the training and validation set. Other methods such as random cut could not be applicable since the entire of the CK+ dataset [5] is already normalized. In the steps of image data preparation we have resized images to 350*350. Hence, all the images are resized to 4848 matrix. by tensor transform then we take the various epochs over 25,50,75,100. And run the same training data over the various epochs to get an idea over how much the accuracy of the image being classified is varying with the epochs.

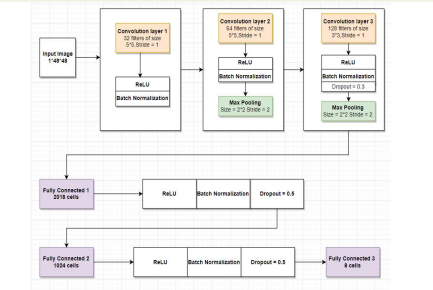
• Given a set of images, design a sequence of convolution and pooling layers with respective activations.

• For each set of input(s), perform forward propagation performing the convolution and pooling operations to find the output.

• Compute the cost using the computed output and actual target of the input data and backpropagate the gradients to update the weights.

Methodology

Network Architecture:



Training / Validation Split:

The following was the split of the given data containing 521 images :

No. of training samples = 416 images

No. of validation samples = 105 images

80% of the data set is used for training and 20% is used for validation.

Hyper-parameters:

The following hyperparameters were tuned using either cross-validation.

Number of layers = 3 (Conv + Pool) layers + 3 fully-connected layers

Number of weights / connections / filter sizes as indicated in above architecture diagram.

Conv layers: 32 (5 x 5) filters -> 64 (5 x 5) filters -> 128 (3 x 3) filters

Pool layers: Max Pooling with 2 x 2 filters with stride 2

(Common choice for natural images)

Number of fully-connected neurons = dimension of the final output from Conv + Pool layer

Batch size = 30 (Set by trial-and-error performances)

Learning rate = 0.01 (Tuned randomly with multiple trials)

Momentum = 0.9 (Tuned randomly for fastest convergence)

Optimization Tricks:

***Cost function* : Cross Entropy Loss (CE)** between the output and target label vector

Stochastic Gradient Descent (SGD) [3] optimizer with momentum was used to achieve more efficient optimization.

Adagrad (Gradient Descent) [1] It adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features. For our case, Adagrad and SGD with different gradients behaves the same resulting in evidence that Adagrad changes learning rate as required. And the results when we used Adagrad also have been extrapolated.

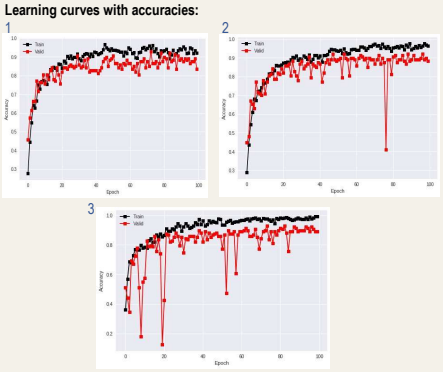
Regularization Strategy:

Batch Normalization [2]**:** Adjusting and rescaling the activations to avoid covariance shift over consecutive layers by normalizing over batch of samples used

***Dropout* :** Randomly ignores certain weights

Results

Learning curves with accuracies:



Final test accuracies with different optimizers

Methods	Final test accuracy (%)
1. With SGD and fixed learning rate	83.464
2. SGD with changing learning rate(after few epochs)	88.188
3. With Adagrad	88.976

Discussions

Why the method pursued performs better than the basic CNN?

Higher Capacity: The CNN used for our model has a higher capacity than normal CNN. The final CNN has multiple neurons and layers and had much higher capacity than the basic CNN.

Representational Power: As the input data samples are images, CNN is a good choice for a model.

Regularization: In the CNN model implemented, regularization was achieved by using **dropouts and batch normalization**. This ensured that the model did not end fitting the noise more than the data.

Optimizer: *With auto learning rate updation method like* ***Adagrad*** *the proposed CNN model is expected to perform better than any other models being evaluated in this project.*

Acknowledgements

The implementation of feature extraction and data collection for the CK+ dataset was done by Anirudh and Naveenkumar. The design of the CNN Model and the data preparation for the same was done by Marulasidda Swamy.The running and capturing of results and graphs was done by MarulasiddanSwamy. The formatting and the contents of the poster were done by Anirudh and Naveenkumar. In all, it was a team effort.

We are grateful to Google Inc. for their generous provision of access to GPUs on the Google Colab platform. It immensely helped us to improve our models with lesser computation time.

References

[1] Duchi, J., Hazan, E., Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12, 2121–2159.

[2] Bower, J.M. & Beeman, D. (1995) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[3] J. Kiefer and J. Wolfowitz Stochastic Estimation of the Maximum of a Regression Function Ann. Math. Statist. Volume 23, Number 3 (1952), 462-466.

[4] Kanade, T., Cohn, J. F., Tian, Y. (2000). Comprehensive database for facial expression analysis. Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), Grenoble, France, 46-53.

[5] Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., Matthews, I. (2010). The Extended Cohn-Kanade dataset (CK+): A complete expression for action unit and emotion-specified expression. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, 94-101.