Static Facial Expression Analysis Using CNN

Jiatao Xiang 1004236613
Xiaohe Gong 1003826782
Bohan Jiang 1004115879

December 2020

Abstract

Human facial expression conveys important information and feedback, so recognizing facial expression has become a very popular field in academia. In this report, we explore and examine the performance of several Convolutional Neural Network (CNN)-based models on the problem of facial expression analysis. We will share the results and effectiveness of our experiments on various techniques and data-processing methods to improve the final accuracy. We focused on training our models on the CK+ and FER-2013 datasets, where our best models achieved 99.1% and 53% test accuracy respectively.

1 INTRODUCTION

Facial Expression Analysis is the problem of identifying and analyzing facial expressions as facial parts change. Automatically recognizing facial expressions is widely applied in fields such as human-computer interaction, computer graphic animation and psychology. The results of detected facial emotions are important feedback for businesses and researchers to improve their work.

Given an image of a person's face, we predict the person's current emotion state (happy, sad, angry, surprise, scared, disgust and neutral) based on the captured facial features. Our models are trained to classify the 7 states as accurate as possible. We've also transferred the model trained for static images into a real-time streaming facial expression detector. The detector serves as an interactive UI, where the webcam streams captured images back to the model and the model outputs predictions and display the possibilities on the UI. Users can also take a selfie with the UI to check the classified emotion.

LITERATURE REVIEW

Many researchers have investigated relevant topics and approached this problem in numerous ways. With these resources, we were able learn about the problem, construct several models and experiment various approaches that we're interested in.

We started by looking at papers that used CNN models. In Pramerdorfer and Kampel's paper on State of the Art [16], we explored the performance and efficiency of different important CNN architectures that are proven to work well on the facial expression recognition problem. We then decided to focus on constructing CNN-based models for our project and gained insights on how the models can be improved.

Nwosu Et al's research paper on Deep CNN model using facial parts [3] is also an important motivation for us. This paper proposed a simple CNN where detected facial parts - eyes and mouth - are cropped and saved as 2 input images, then used as input to a 2-channel CNN. Eyes and mouth images go through one channel each and finally the two channels converge in a fully-connected layer to produce the final prediction. Since eyes and mouth convey important information about a person's emotion, using facial parts as input would allow the model to focus on useful information. This neural network is fast to train and efficient in memory as the feature extraction step is mostly done outside of the CNN.

We've also gained a lot of insight from Li and Deng's paper [12] discussing the effectiveness and performance of different FER datasets. With their findings, we learned that FER-2013 is a large but noisy dataset fetched using Google API. The human accuracy of FER-2013 is around 65% and the current best models achieves accuracy at around 67 - 75%. One of the other smaller, lab-controlled dataset CK+ is also a good dataset to use, as the images are posed and spontaneous. We decided to focus on FER-2013 and CK+ since the two datasets would bring both diversity and accuracy to our model.

2 Methods, Results, and Experiments

We will now discuss about our results on the two datasets we focused on: CK+ [4] and FER-2013 [9]. We splitted both datasets by 70% training and 30% validation.

The Extended Cohn-Kanade Dataset (CK+) is one of the most popular dataset for this problem. It is a lab-controlled dataset with 981 posed, spontaneous images of size 48×48. Because of the quality and size of CK+, models tend to achieve very high accuracy. However, the trained models have low accuracy when testing with real-world settings, because those images are much more noisy.

FER-2013 is a much larger dataset with 35887 images of size 48×48. It is collected automatically using the Google Image API. As a result, many images are not posed and therefore challenging to achieve high accuracy. For reference, the human accuracy on this dataset is around 65%. We spent most of our time trying to improve our model on FER-2013, as the dataset is diverse and closer to real-world settings.

A. CNN by Facial Parts

Our first model is inspired by the CNN architecture discussed in [3]. We implemented their architecture in Pytorch and added improvements based on our own understandings. The model architecture is shown in Figure 2.1.

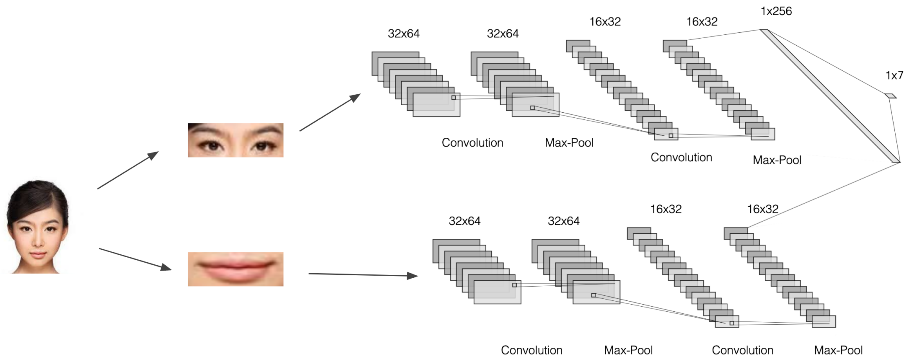


Figure 2.1: A simple CNN architecture with facial parts as input

DISCUSSION. As we can see, this architecture is very intuitive. It is composed of 2 CNN channels, each with 2 Conv2D layers (with kernel size 3) and 2 Max-Pool layers (with kernel size 2). As human, we know that eyes and mouth are important facial features for emotion changes, so we would expect this information to help the model to achieve better accuracy. Therefore, we pass the eyes and mouth components into each of the CNN channel. The result of the 2 channels converge at the final fully-connected layer and output the most probable facial expression out of the 7 classes. Since this model is small, it is fast to train and cheap in the number of parameters.

PRE-PROCESSING. Since we are using facial parts as input to the model, we first pre-processed the images into the eyes and mouth components of size 32×64 each. The face and facial landmarks are detected using `dlib`'s pre-trained frontal face detector [8][7]. We wrote our own `util` class to crop and resize the image with the coordinates of eyes and mouth.

EXPERIMENTS & RESULTS. We experimented with tuning different parameters including kernel size and number of channels on the CNN. With this model, we achieved test accuracy of 91% on the CK+ dataset and 42% on the FER2013 dataset.

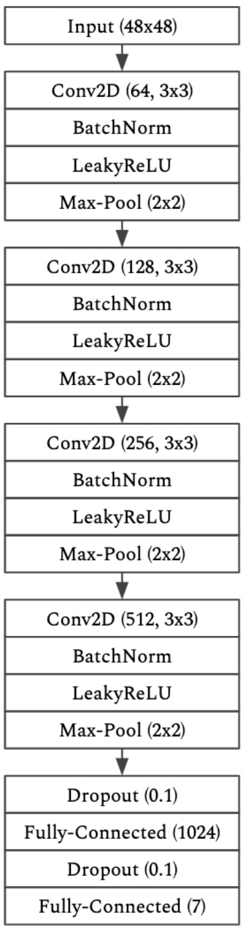
In an attempt to improve the accuracy, we added Histogram of Oriented Gradients (HOG) descriptors to provide more information to our model. We modified the model to concatenate features from the HOG descriptor in the last fully-connected layer. The test accuracy of CK+ dataset increased to 99%, while the FER-2013 dataset did not have much improvement.

We also implemented data augmentation by applying random rotations and flips to the image. However, this did not seem to improve the accuracy as well.

Based on our research, we know the models that achieved high accuracy on FER-2013 are all very large (E.g., VGG-Net). Therefore, we then tried to make our model deeper by adding more convolution layers. However, the accuracy was still not high at around 41%. We concluded that using eyes and mouth as input might not be a good idea for FER-2013. This is because many images are not posed, so the cropped out eyes and mouth images may not be as useful as we imagined. Consequently, we lost important features and had negative impact on the model.

B. Our Original CNN Model

After knowing that we should deepen the CNN model, we looked into the VGG-16 architecture [6] and some of the open-source projects [10]. We designed a deep CNN model (shown on the right) using our knowledge and insights from existing best models. For this model, we are using the original image (not eyes and mouth) as input.



Pre-processing. We used data augmentation and data centering in this step. We tried to apply random rotations, flips and transformations, but it did not really improve the result. We also tried to center the pixels by subtracting the mean from each image. This improved the model only slightly, possibly because we already have BatchNorm layers.

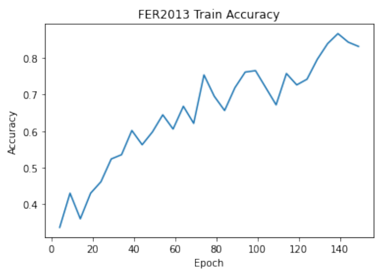
Enhancements We used weight initialization in this model. We chose to initialize the CNN's weights using the Uniform distribution, which lowered the loss and improved the accuracy slightly. Based on our research [5], this would prevent the exploding/vanishing gradient problem during training. During our experiment, training converged much faster.

We also tried to integrate our model with HOG descriptors again as suggested by some results in [10]. Unfortunately, it yielded worse accuracy. We believe the model did not capture the HOG features properly possibly because the model over-fitted with too much information.

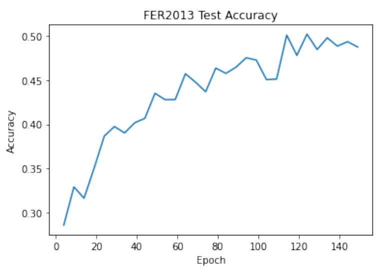
Experiments & Results.

- After some experiments, we found that using `SGD` optimizer with momentum 0.95 and weight decay `5e-4` works best with our model.
- The `BatchNorm` layers improved the test accuracy by about 10%.
- Using `LeakyReLU` prevents the dying ReLU problem.
- Using this model, we kept the original full 48×48 image as input, which improved the FER-2013 test accuracy quite a lot.
- We trained 150 epochs with mini-batches of size 128.

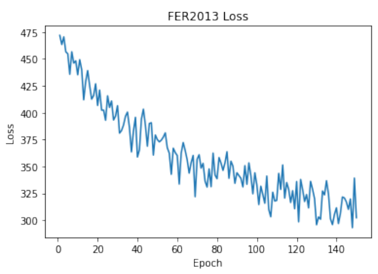
The test accuracy is now 53% for FER-13, and 99% for CK+.



(a) Train Accuracy



(b) Test Accuracy



(c) Loss

C. Support Vector Machine (SVM) Classifier

After achieving 53% accuracy on the CNN model, we explored the empirical, traditional SVM method. SVM is a popular approach in pattern recognition tasks such as facial action recognition. It attempts to maximize the hyper-plane that separates positive and negative observations. For the FER problem, we will need to perform a multi-class decision (E.g., by deciding happy vs not-happy).

Method. We implemented a simple SVM classifier using sklearn's built-in functions [17]. We used the one-vs-one scheme with Radial Basis Function (RBF) as kernel. As suggested by FER researches using the SVM method[13], RBF kernels significantly outperforms the others. For input, we just used the grayscale raw pixels. After some parameter tuning, we achieved 85% test accuracy on CK+ and 44.7% test accuracy on FER-2013 datasets.

Discussion. As we can see, the results of the SVM Classifier is not as good as the 53% accuracy by CNN models. This is probably because FER is a multi-class problem, but SVM performs well on small dataset and binary classifications [15]. It also took a very long time to train (around 2 hours) with the FER-2013 dataset.

As suggested by research in [1] [18], integrating SVM with the Principal Component Analysis (PCA) and Local Binary Pattern (LBP) algorithms would improve the result. Researchers have also attempted a SVM-AdaBoost facial expression recognition system [2], which achieved high accuracy at 97.57% for the JAFFE dataset. These are some enhancements we could try in the future.

3 Discussion

Result

The final accuracy is summarized in table 3.1.

Table 3.1: Model Results

	CK+		FER-2013	
Method	Training Accuracy	Test Accuracy	Training Accuracy	Test Accuracy
CNN Model A	98%	91%	39.3%	42.3%
CNN Model A + HOG	98%	99%	45.2%	42.6%
CNN Model B	99%	**99.1**%	87.4%	**53.2**%
CNN Model B + HOG	99%	99%	88.6%	51.2%
SVM Classifier	94.6%	85.1%	65%	44.7%

For CK+ dataset, our CNN models all had descent accuracy at above 90%. The SVM model performed not as well, but still quite good. For FER-2013 dataset, our best model achieved test accuracy of 53.2%. Interestingly, HOG improved CNN Model A slightly but did not improve CNN Model B. This is probably due to the simple structure of Model A is missing a lot of features, and HOG can help with it. However, such features are already detected for Model B, as it's larger and deeper.

Furthermore, we found out that certain emotions have a lot fewer training data in FER-2013, for example, there're only 547 images are 'disgust', whereas 'happy' has 8989 images, so our model performs very well for ‘happy’, but not as well for ‘disgust’.

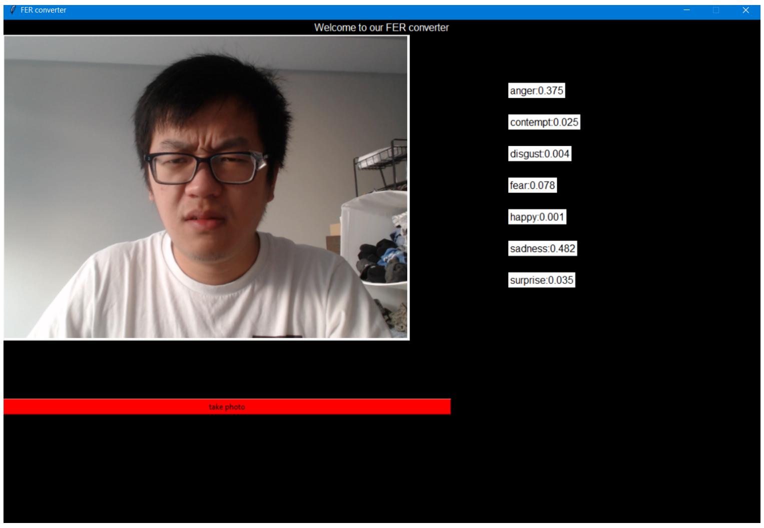
Even though our model’s result is not as good as the best CNN models such as VGG-16 [11] (FER-2013 accuracy ≈ 73%), our model accuracy is descent when testing on our interactive UI.
Compared with models like VGG-16, our model takes a lot less time to train:

- The CK+ dataset converged in less than a minute
- With Google Colab’s GPU, CNN Model A and Model B training converged within 30 minutes when we were training for 150 epochs with batch size 128.

We also spent few days trying to reproduce the results of VGG-16, but due to hardware constraints, we were not able to train models of that size. This is why we proceeded with our customized model.

INTERACTIVE FER UI

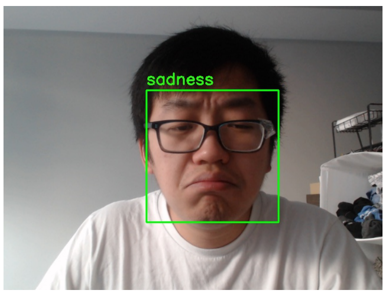
Finally, our final CNN model B is piped into a FER analyzer that users can take a selfie and test out our model with. This interactive UI is designed and implemented by ourselves using OpenCV libraries [14].



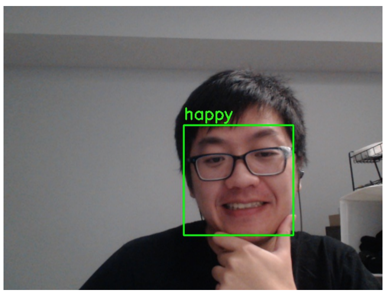
```
anger:0.375
contempt:0.025
disgust:0.004
fear:0.078
happy:0.001
sadness:0.482
surprise:0.035
```

When the selfie is saved, we use the same facial detector from `dlib` and pass the captured facial image into our model. Finally, the classification likelihood output are sent back to the UI.

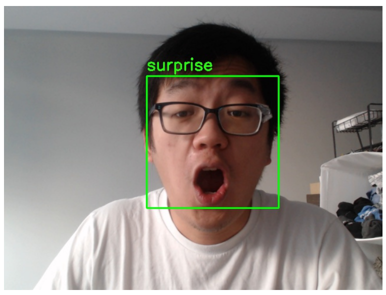
Some of the successful results are shown below. Depending on the background and image noise, the result can be pretty wrong, but we had a lot of fun testing our model.



(a) Sad



(b) Happy



(c) Surprise

4 Conclusion

In this project, we focused on using CNN models to tackle the problem of facial expression analysis. We compared the results of 2 CNN models against the result of SVM Classifier along with several enhancements that we think would improve our model. These enhancements include integrating models with HOG descriptor, data-augmentation that applies random image transformations, weight initialization, and parameter tuning. Finally, we built a interactive UI that allows users to take selfie and test our model. Due to the nature of FER-2013 dataset, it is very challenging to have a model that performs better than 65% (the human accuracy). Our medium-sized CNN model was able to achieve 53% test accuracy on FER-2013, and the training time and costs are relatively efficient compared to the best accuracy models by others [16].

5 Author Contributions

Each one of us was responsible for relative equal amount of research and work:

- Jiatao Xiang - Research, proposal write-up, brainstorm, design and implement CNN Model base structures, implement data-loaders, build and experiment with VGG-16 size models, build CNN facial by parts model A, build interactive UI from scratch
- Xiaohe Gong - Research, proposal write-up, train and fine-tune CNN by parts model A, build and train CNN Model B, pre-process FER2013 datasets, report write-up, presentation write-up
- Bohan Jiang - Research, proposal write-up, implement face and facial landmark detection, implement image pre-processing and pre-process CK+ datasets, integrate and experiment HOG descriptors, implement SVM classifier

6 Code

Quick demo on results and FER Interactive UI

`https://youtu.be/WO97TrLZTqA`

`https://drive.google.com/file/d/1Xr8dEYBNQW8HhDFEjfMvNZAGv32GZgvw/view?usp=sharing`
(backup url)

Code zip file

`https://drive.google.com/file/d/1OclKBPyBFd69h94Vb3-HGdtyrRiVsLVn/view?usp=sharing`

REFERENCES

[1] M. Abdulrahman and A. Eleyan. “Facial expression recognition using Support Vector Machines”. In: *2015 23nd Signal Processing and Communications Applications Conference (SIU)*. 2015, pp. 276–279. DOI: 10.1109/SIU.2015.7129813.

[2] Ebenezer Owusu Et al. *An SVM-AdaBoost facial expression recognition system*. URL: https://link.springer.com/article/10.1007/s10489-013-0478-9.

[3] Lucy Nwosu Et al. *Deep Convolutional Neural Network for Facial Expression Recognition using Facial Parts*. URL: https://sceweb.uhcl.edu/xiaokun/doc/Publication/2018/ICPI2018_Lucy.pdf.

[4] *CK+ dataset for facial expression recognition*. URL: https://www.kaggle.com/shawon10/ckplus.

[5] James Dellinger. *Weight Initialization in Neural Networks: A Journey From the Basics to Kaiming*. URL: https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79.

[6] Poonam Dhankhar. *ResNet-50 and VGG-16 for recognizing Facial Emotions*. URL: http://ijiet.com/wp-content/uploads/2019/08/18.pdf.

[7] *dlib documentation*. URL: http://dlib.net/python/index.html.

[8] *Facial landmarks with dlib, OpenCV, and Python*. URL: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/.

[9] *fer2013*. URL: https://www.kaggle.com/deadskull7/fer2013.

[10] Amine Horseman. *Facial expression recognition using CNN in Tensorflow*. URL: https://github.com/amineHorseman/facial-expression-recognition-using-cnn.

[11] H. Jun et al. “Facial Expression Recognition Based on VGGNet Convolutional Neural Network”. In: *2018 Chinese Automation Congress (CAC)*. 2018, pp. 4146–4151. DOI: 10.1109/CAC.2018.8623238.

[12] Shan Li and Weihong Deng. *Deep Facial Expression Recognition: A Survey*. URL: https://arxiv.org/pdf/1804.08348.

[13] Philipp Michel and Rana El Kaliouby. *Real Time Facial Expression Recognition in Video using Support Vector Machines*. URL: https://www.cs.cmu.edu/~cga/behavior/FER-SVM-ICMIpaper.pdf.

[14] OpenCV. *Capture Video from Camera*. URL: https://docs.opencv.org/master/dd/d43/tutorial_py_video_display.html.

[15] P. Jonathon Phillips. *Support Vector Machines Applied to Face Recognition*. URL: https://www.researchgate.net/publication/2414722_Support_Vector_Machines_Applied_to_Face_Recognition.

[16] Christopher Pramerdorfer and Martin Kampel. *Facial Expression Recognition using Convolutional Neural Networks: State of the Art*. URL: https://arxiv.org/pdf/1612.02903v1.pdf. (accessed: 12.10.2020).

[17] scikit-learn. *sklearn.svm.SVC*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[18] Yichuan Tang. *Deep Learning using Linear Support Vector Machines*. URL: https://arxiv.org/pdf/1306.0239.pdf.