Learning Face Emotion with Deformable Model

Yu Hou, Xueer Li, Zongnan Bao, Xiaoyang Yu, Felix Zhang
Department of Computer Science. University of California, Los Angeles
{yuhou316, lixueer, zb3, xiaoyangyu, felix44}@g.ucla.edu
UID:{105711952, 805498731, 405711837, 005726398, 905203031}

Abstract

Face detection and emotion recognition have been investigated deeply in the past few decades because they are essential for identification. This project presents facial landmark detection using a deformable part model based on 2D images and extracted face shape and other necessary features to predict facial emotion. Compared with the previous models on emotional detection, this project explores various convolutional neural network frameworks and deformable models to increase emotion recognition accuracy. The experimental results indicate that the proposed methods have good performance for 2D face detection and emotion recognition. Furthermore, we present an algorithm for 2D face emotion reconstruction based on deformable models and demonstrate the effectiveness and strength of our algorithm both qualitatively and quantitatively.

1. Introduction

Since the last several decades, the study of the face and its characteristics has been a hot topic of research. All algorithms are nevertheless challenged by issues such as pose variation, lighting conditions, etc. Face recognition and emotion detection are two of the most common uses of recognition systems, and several algorithms have been developed to address these issues. As a critical component of current authentication and identification systems, the system's accuracy is critical.

There are some effective approaches for recognizing faces and facial expressions, like such as Deep-emotion [8] and the FaceChannel [1] etc. Happy, Anger, Disgust, Fear, Sadness, and Surprise are among the expressions that are examined for recognition. When applied to the most extensively used databases, the strategies are thought to have a very high recognition rate. These methods employ a hybrid algorithm that takes into account both holistic and local characteristics in order to improve recognition rates and performance.

In order to achieve better performance, our project used a deformable model based on 2D images to recognize facial landmarks and extract face shape and other essential data to predict facial emotion. To improve emotion identification accuracy, our project integrated the convolutional neural network with deformable models, as opposed to earlier models on emotional detection. The experimental findings showed that the suggested technique is effective at detecting 2D faces and recognizing emotions. We also demonstrated its usefulness and strength both subjectively and statistically. In the future, we plan to provide a deformable model-based approach for 2D face emotion reconstruction.

The rest of the report is organized as follows. Section 3 introduces some current face emotion detection models. Section 4 describes the method we used for the project. We will demonstrate our training and testing results of the deformable model, as well as how to enhance emotion recognition with deformable models in later sections. Finally, the conclusion of our project will be discussed.

2. Related works

2.1. Deep-Emotion

In Deep-Emotion [8], Minaee et al. provided a novel framework for face expression recognition. The authors believe paying attention to specific areas is critical for recognizing facial expressions, and that neural networks with fewer than ten layers may compete with much deeper networks in this regard.

To categorize the underlying emotion in face photos, they used an end-to-end deep learning system based on an attentional convolutional network. Adding extra layers, increasing gradient flow in the network, or better regularization are frequently used to improve a deep neural network, especially for classification problems with a high number of classes. Due to the small number of classes in facial emotion detection, they showed that employing a convolutional network with fewer than 10 layers and attention may yield promising results, outperforming state-of-the-art models for numerous databases. They also used a visualization approach to emphasize the most important sections of face

photos for recognizing distinct facial expressions.

2.2. FaceChannel

In 2020, Barros et al. presented FaceChannel [1], a neural network with many fewer parameters than traditional deep neural networks. This model is an improved version of the MultiChannel Convolution Neural Network. The FaceChannel is a light-weight convolution neural network that has been trained from the ground up to produce state-of-the-art FER outcomes. To assess the model, they ran a series of tests using various facial expression datasets and compared the results to the existing state-of-the-art. When it comes to modeling affect from facial expressions, each dataset has its own set of constraints. FaceChannel outperformed the state-of-the-art solutions published on these datasets.

3. Method

3.1. Making face feature contours from images with deformable model

Basically, we used Menpo library to gain contours of face features. Menpo library is a set of set of BSD licensed Python frameworks and associated tooling that provide end-to-end solutions for 2D and 3D deformable modeling [4]. Inside the Menpo project, we mainly use Active Appearance Model inside the Menpofit. The theories below all come from the introduction part of active appearance model inside the Menpo. Our team does not have any right or contribution to the following theories.

Active Appearance Model is a model for the shape and new object class, and it is also a statistical deformable model. Basically, the input of this models is a list of images with $images = [image1, image2, image3, ...]$. Corresponding to every image, we also have the landmark/feature points $p = [(x1, y1), (x2, y2), ...]$. Every image should have the same number of landmark points and typically the number is 68.

Active Appearance Model is trained by doing following. For shape model, we used Generalized Procrustes analysis and Principal Component Analysis to get s, U_s where s is the mean shape vector. Therefore, a new shape can be expressed as $s_{new} = s + U_s p$ where p is the vector of shape parameters of a new image [3].

For appearance model, We use the features function F defined inside holistic_features to get the features I, and then we warp(W) the images based on these features to the reference_shape $a = F(I)W(p)$, and after that, we use Principal component analysis where we can get a, U_a where $U_a \in R^{m*m}$ is the orthonormal basis of m eigenvectors and $a \in R^{M*1}$ is the mean appearance vector [3] [6]. Then based on the introduction, we can get a new appearance instance by $a_{new} = a + U_a c$ where c is the vector of appearance parameters.

The cost function of AAM is

$$argmin||F(I)W(p) - a - U_a c||^2$$

where p is the shape parameters, I is the feature points, a and U_a is getting from appearance model, c is the appearance parameters

PatchAAM is the active appearance model in patch base. That is a change for warping function. Inside warping as a whole image, we make the warping function to warp the rectangular patches centered at the feature points [3].

We used Lucas-Kanade Optimization [5]. That is basically a gradient descent method where shape parameters update in every iteration until the change is small.

$$W(p) = W(p) \circ W(\triangle p)^{-1}$$

3.2. Establishing Deep-Emotion Baseline

We run Deep Emotion over various datasets including FER_2013, FER_CLKPLUS, FER_CKPLUS_KDEF, and CKPLUS. FER_2013 is a series of image grayscale faces in a variety of poses. Due to the dynamic poses, the data is difficult and poses challenges to models. FER_CKPLUS_KDEF is a cleaned version of FER, CK, and KDEF emotion datasets.

We explore over various hyperparameters, including dropout, weight-decay, training schedulers, dropout and number of dropout layers, learning rate. We also explore various derivatives of Deep Emotion, varying model width and depths to explore how Deep Emotion performs as a baseline.

3.3. Exploring Usefulness of Deformable Convolution and Feature Contours

We introduce Deformable Convolution [2] and Feature Contours [7] into the architecture of Deep Emotion. We run an ablation study with Deformable Convolutions and Feature Contours on an annotated subset of FER_CKPLUS_KDEF to determine the effect of the models on model accuracy. Note that the annotated subset is much smaller than the actual FER_CKPLUS_KDEF due to computation limitations. We arbitrarily chose which data to annotate as well.

3.4. Training Details

We train with Google Colab and AWS EC2. On Colab we typical receive a K80 instance. For our AWS EC2 instance, we trained with a p2.xlarge instance before converting into a p3.2xlarge instance. The p3.2xlarge instance provides a Tesla V100, which is able to train a shallow Deep-Emotion on our subset dataset in around an hour wall-clock time. Deeper iterations on the larger dataset obviously have

larger wall-clocks. We estimate each run to be around 5 hours with our deepest network on the full dataset. Due to financial limitations, we generally train on the K80 provided by Colab, resulting in around a 3x increase in training time.

4. Experiments

4.1. Add contour to images

We used the LFPW [7](Labeled Face Parts in the Wild) dataset to train the AAM model from Menpo. This dataset consists of 1,432 faces from images downloaded from the web using simple text queries on sites such as google.com, flickr.com, and yahoo.com. Meanwhile, the dataset includes the 68 landmark points for every image. We changed the grey scale, landmark proportion and size of each image from training dataset to build and fit their respective AAMs. So, the first step, we loaded a set of images, cropping them and converting them to greyscale.

Furthermore, igo features are computed on an feature image that was already obtained by computing igo features. Then we are ready to to build and fit a patch-based AAM: the appearance representation consists of small patches extracted around each of the landmark points. i.e. rectangular patches are extracted around the landmark points. We used patch-based AAM builder, PatchAAM from Menpo and trained such an AAM using IGOs with double angles as feature representation. After trained AAM, we use the algorithms of the LucasKanadeAAMFitter to fit the trained Patch-Based AAM. We can see from Figure 3 and Figure 4 that, the final landmarks are more accurate than the initial one.

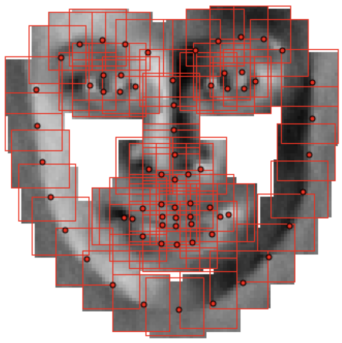

Figure 1. Patch Based AAM Widget

```
[[ 94.74282266  43.8829351 ]
 [117.42948266  44.7984122 ]
 [140.38310434  47.24959671]
 [162.96926716  50.78164366]
 [183.82356482  58.56867651]
 [201.63910363  72.27079174]
 [216.99080184  89.0858746 ]
 [229.99888373 108.72289806]
 [233.40911455 132.25455091]
 [229.49188579 155.7109645 ]
 [215.53790423 175.60622887]
 [198.97244494 192.97530237]
 [179.82499489 206.36345727]
 [157.78536827 213.42754301]
 [134.56414035 215.99772383]
 [112.06932115 217.0710115 ]
 [ 89.89652393 217.27780414]
 [ 75.40222387  59.21295157]
 [ 65.82530669  69.9615757 ]
 [ 63.22237386  84.30710175]
 [ 65.03102642  99.13218728]
 [ 70.25829494 113.44854211]
 [ 69.77277952 145.38363803]
 [ 64.17813018 160.23217216]
 [ 62.3519156  174.83213677]
 [ 64.83241578 189.19606171]
 [ 73.47691935 199.93400886]
 [ 88.04544167 130.48249939]
 [102.67351474 130.74322491]
 [117.33715497 130.96793556]
 [132.42743596 131.25966688]
 [139.47480924 112.08290962]
```

Figure 2. Result Point Snippet

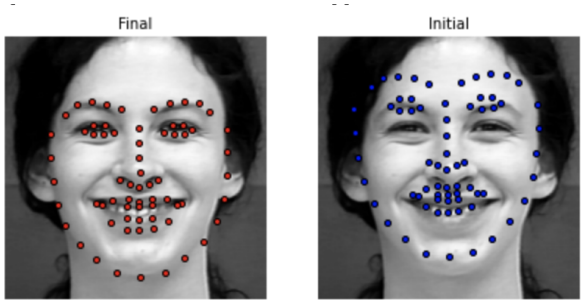

Figure 3. Comparison between final and initial landmarks

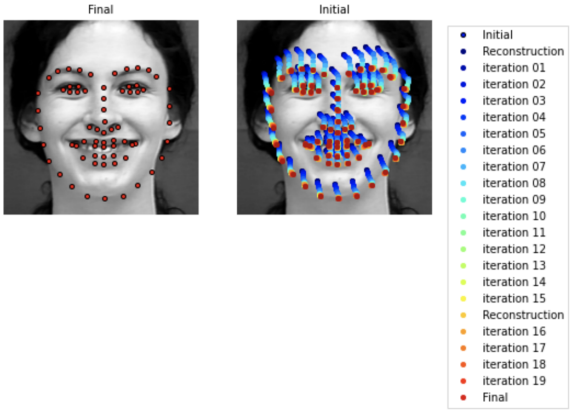

Figure 4. Landmarks in each Iteration

After we trained the LFPW dataset, we are able to see the landmark points on each image. We can see this happy face with the landmarks trained by AAM. we shown in the Figure 5. There would be sixty-eight points on each face: seventeen of them are Jawline contour; ten of them are eyebrows contour; nine of them are nose contour; ten of them

are eyes contour and twenty-two of them are mouth contour. For each feature, we extracted the points and add a line between two points adjacent to each other and created a contour picture of a human face. In this happy face example, we can see the final result in Figure 6.

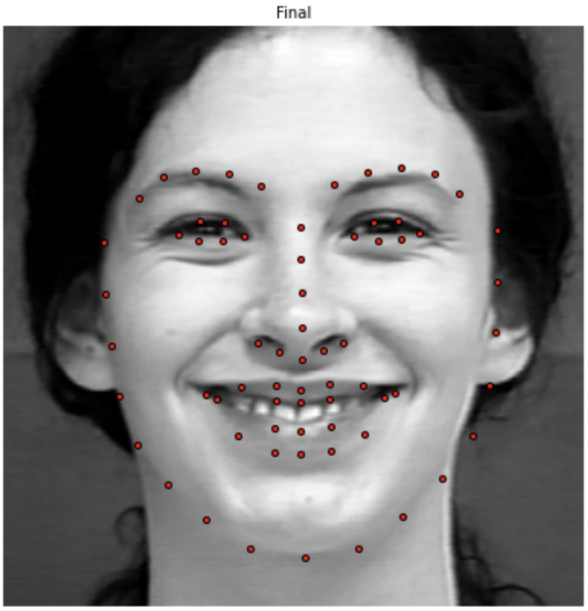

Figure 5. Happy Face Example with landmarks

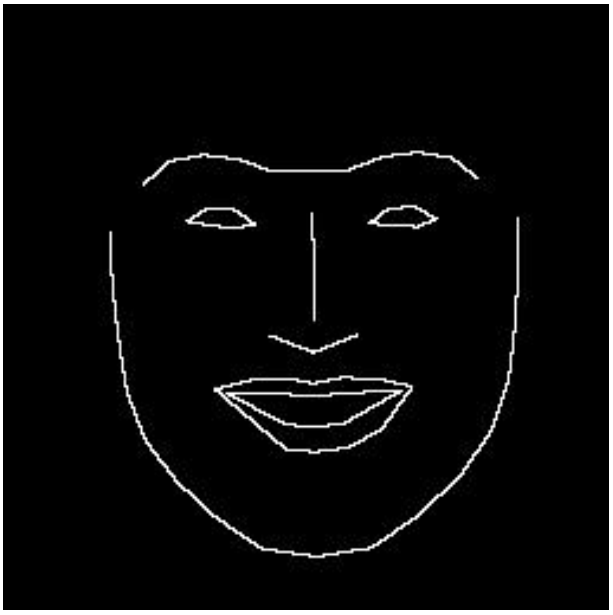

Figure 6. Happy Face Example Contour Result

4.2. Baseline Experiments for Deep Emotion

We explore various hyperparameters and architecture choices across different datasets as shown in **Table 2**. We see that Deep Emotion is able to achieve SOTA on CK_PLUS_256 with deformable convolution. One thing to note is that for our FER_CKPLUS dataset, VGG achieves a higher val acc, however has 1000 times more model parameters than Deep Emotion with Deformable Convolutions, which achieves a similar val accuracy.

Also note that for CK_PLUS_256, some of our runs perform very well due to some datasets having a perfect sequence of 3 frames, which we split into train,test,val. Therefore the val is very similar to the train and the test.

Also note that FER_2013 appears to be much more difficult. Upon further inspection we note that the dataset itself has a variety of difficult poses as well as some that are indeterminate. Human accuracy was around 70% for the annotators of the dataset in Kaggle.

In Figure 7 we see a example run of our annotated dataset when we train with the train and val losses and accuracies. Notice this run could be run a bit further than 100 epochs, but to compare models fairly we did a fix epoch. Due to the small dataset size of FER_CKPLUS, we used extensive dropout, which could explain the difficult of convergence especially when it nears the plateau.

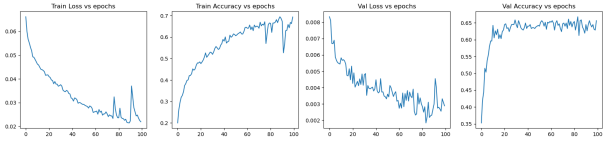

Figure 7. Our annotated FER_CKPLUS dataset we use for testing contour with Deep Emotion model. For each emotion top is image in greyscale, bottom is contour.

We note that the plateau leads not non-convergence unless we set a lower bound to the learning rate, and that sometimes our dropout is too strong and needs reducation. We also explored various data augmentations, since especially for our annotated dataset, our dataset is much smaller.

Some datasets are 224x224 rather than 48x48, and we deepen the network accordingly to accomodate that. We notice that wider networks seem to perform better, and that really deep networks tend to overfit on our dataset (especially sometimes our training accuracy is much higher than our validation).

4.3. Ablation Experiments for Deformable Convolution and Feature Contours

We perform an ablation to see if our Contour and Deformable Convolution really improves our Base Deep Emotion model. We observe that the Contour doesn't seem

to change our validation accuracy by much, but the Deformable Convolution boosts it a bit.

In the future, it would be helpful to run an ablation with more hyper parameters. We also observed that it might be interesting to use the mask as attention (or just pass it into our spatial transformer network to generate the transformation).

Architecture	Training Accuracy	Validation Accuracy
Base Deep-Emotion	68.37	62.18
Contour + DE	69.15	60.325
Deform. Conv + DE	69.46	**66.82**

Table 1. Ablation study with annotated dataset. Hyperparams are 100 epochs, 0.4 dropout, Adam Optimizer, 1e-4 lr, 1e-2 weight decay. More grid search for hyper-parameter is needed, but initial results point to contours not being particularly helpful, but Deform. Conv being useful.

References

[1] Pablo Barros, Nikhil Churamani, and Alessandra Sciutti. The facechannel: A light-weight deep neural network for facial expression recognition. *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, Nov 2020. 1, 2

[2] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017. 2

[3] Menpo developers. Active appearance model, 2018. 2

[4] Menpo developers. The menpo project, 2018. 2

[5] G. Tzimiropoulos E. Antonakos, J. Alabort-i-Medina and S. Zafeiriou. Feature-based lucas-kanade and active appearance models. vol. 24, no. 9, pp. 2617-2632, 2015. 2

[6] J. Alabort i Medina and S. Zafeiriou. "a unified framework for compositional fitting of active appearance models. 2

[7] LLC. Kriegman-Belhumeur Vision Technologies. Labeled face parts in the wild, 2011. 2, 3

[8] Shervin Minaee and Amirali Abdolrashidi. Deep-emotion: Facial expression recognition using attentional convolutional network, 2019. 1

Appendix

Figure 8. Our annotated FER_CKPLUS dataset we use for testing contour with Deep Emotion model. For each emotion top is image in greyscale, bottom is contour.

Experiments of Architecture and Parameters								
Architecture and Dataset			Hyperparameters					
Model	Model Details	Dataset	Epochs	Optimizer	Batch Size	Train Acc%	Val Acc%	Data Aug
Deep-Emotion	Vanilla, baseline	FER_CKPLUS	100	Adam, lr=1e-4, weight_decay=1e-4	64	64.853	62.573	RandomFlip
Deep-Emotion	Wider, channel: 1064	FER_CKPLUS	100	Adam, lr=1e-4, weight_decay=1e-4	64	85.233	64.375	RandomFlip
Deep-Emotion	Deeper, two more conv layers	FER_CKPLUS	100	Adam, lr=1e-4, weight_decay=1e-4	64	67.977	58.509	RandomFlip
Deep-Emotion	Wider + Deform. Conv (2 lyrs)	FER_CKPLUS	100	Adam, lr=1e-4, weight_decay=1e-4	64	89.101	**68.561**	RandomFlip
VGG	Vanilla VGG(1,000x param)	FER_CKPLUS	100	Adam, lr=1e-4, weight_decay=1e-4	64	98.587	**70.241**	RandomFlip
Deep-Emotion	Wider, channel: 1064	CK_PLUS_256 (48x48)	100	Adam, lr=1e-4, weight_decay=1e-4	64	100	72.727 [80+]	RandomFlip
Deep-Emotion	Wider, channel: 10→ 64	CK_PLUS_256 (256x256)	100	Adam, lr=1e-4, weight_decay=1e-4	16	98.639	93.939	Flip and Crop
Deep-Emotion	Wider, channel: 10→64 (10 fold)	CK_PLUS_256 (256x256)	100	Adam, lr=1e-4, weight_decay=1e-4	16	99.660 (10-fold)	92.055 (10-fold)	Flip and Crop
Deep-Emotion	Wider + Deformable Conv (2 lyrs)	CK_PLUS_256 (256x256)	100	Adam, lr=1e-4, weight_decay=1e-4	16	100 (10-fold)	95.095 (10-fold)	Flip and Crop
Deep-Emotion224	Deep_Emotion224	CK_PLUS_256 (256x256)	200	Adam, lr=5e-5, weight_decay=1e-4	16	99.660 (10-fold)	94.186 (10-fold)	Flip and Crop
Deep-Emotion	Baseline Deep-Emotion	CK_PLUS_256 (48x48)	300	Adam, lr=1e-4, weight_decay=1e-4	32	86.646 (10-fold)	85.597 (10-fold)	Flip and Crop
Deep-Emotion	Baseline Deep-Emotion	FER_2013	100	Adam, lr=1e-4, weight_decay=1e-4	64	66.571	51.212	None
Deep-Emotion	Baseline Deep-Emotion	FER_2013	100	Adam, lr=1e-4, weight_decay=1e-4	64	58.783	54.611	RandomFlip
Deep-Emotion	Baseline Deep-Emotion (Closest to paper)	FER_2013	500	Adam, lr=5e-3, weight_decay=1e-4	64	-	Loss Explode	RandomFlip
Deep-Emotion	Baseline Deep-Emotion (reordered RELU)	FER_2013	100	Adam, lr=1e-4, weight_decay=1e-4	64	60.256	54.249	RandomFlip
Deep-Emotion	Baseline Deep-Emotion (reordered RELU)	FER_2013	500	Adam, lr=1e-4, weight_decay=1e-4	64	69.563	54.444	RandomFlip
Deep-Emotion	Baseline Deep-Emotion (reordered RELU)	FER_2013	500	", ReduceLROnPlateau	64	-	-	RandomFlip
Deep-Emotion	Baseline + Deform Conv (x2)	FER_2013	100	Adam, lr=1e-4, weight_decay=1e-4	64	61.754	**56.617**	RandomFlip Jitter
Simple CNN	3Conv + 2FC, simple form	FER_2013	100	Adam, lr=1e-4, weight_decay=1e-4	64	60.782	55.782	Flip and Jitter
DeepEmotion	Baseline + Deformable Conv (x2)	FER_2013	100	Adam, lr=1e-4, weight_decay=1e-2	64	59.915	56.590	Flip and Jitter
DeepEmotion	Deep_Emotion224	CK_PLUS_256	200	Adam, lr=1e-4, weight_decay=1e-2	64	99.966 (10-fold)	94.517 (10-fold)	Flip and Crop
Deep-Emotion	Deep_Emotion224 w/ 2 dropout (0.5)	CK_PLUS_256	200	Adam, lr=1e-4, weight_decay=1e-2	64	58.443	91.126 (10-fold)	Flip and Crop
Deep-Emotion	Deep_Emotion224 w/ 1 dropout(0.8)	CK_PLUS_256	200	Adam, lr=1e-4, weight_decay=1e-2	64	80.667 (10-fold)	89.015 (10-fold)	Flip and Crop
Deep-Emotion	Deep_Emotion224 w/ 2 dropout (0.2)	CK_PLUS_256	400	Adam, lr=1e-4, weight_decay=1e-2	64	90.927 (10-fold)	**96.932** (10-fold)	Flip and Crop
Deep-Emotion	Deep_Emotion,wider,Deform,2 dropout(0.2)	CK_PLUS_256	200	Adam, lr=1e-4, weight_decay=1e-2	64	89.092 (10-fold)	95.133 (10-fold)	RandomFlip
Deep-Emotion	Deep_Emotion,wider,Deform,2 dropout(0.2)	CK_PLUS_256	200	",ReduceLRonPlateau(pat.=15)	64	88.175 (10-fold)	96.628 (10-fold)	RandomFlip
Deep-Emotion	Deep_Emotion,wider,Deform,2 dropout(0.2)	CK_PLUS_256	200	ReduceLRonPlateau(20,lr=1e-7)	64	89.058 (10-fold)	**96.648** (10-fold)	RandomFlip
Deep-Emotion	Deep_Emotion,wider,Deform,2 dropout(0.2)	CK_PLUS_256	200	ReduceLRonPlateau(20,lr=1e-7)	64	70.473 (10-fold)	95.417 (10-fold)	RandomFlip
Deep-Emotion	Deep_Emotion,wider,Deform,2 dropout(0.2)	CK_PLUS_256	200	ReduceLRonPlateau(20,lr=1e-7)	64	92.490 (10-fold)	90.237 (10-fold)	RandomFlip

Table 2. We explore various hyperparameters and Architectures in our experiments. All Convolution and Fully Connected layers are appended with a batchnorm, we iterate off of vanilla Deep Emotion.