Kaggle Competition : Facial Keypoints Detection

- **Yunqiu Xu (z5096489), Junrui Chen (z5124037), Shuo Yang (z5103377), Shunan Shen (z5050880)**

---

1. Introduction

Detecting keypoints in a given face image would act as a fundamental part for many applications, including facial expression classification, facial alignment, tracking faces in videos and also applications for medical diagnosis. We choose a related Kaggle competition[1] as our project, the objective is to predict keypoint positions on face images accurately and fastly.

2. Dataset Description

We are given a training set with 7049 images, each row contains (x,y)-pair as the location of 15 keypoints , and image data as row-ordered list of pixels $(96 * 96, [0, 255])$. The test data consists of 1783 test images. Following is part of training set:

	left_eye_center_x	left_eye_center_y	right_eye_center_x	right_eye_center_y	left_eye_inner_corner_x	left_eye_inner_corner_y
0	66.033564	39.002274	30.227008	36.421678	59.582075	39.647423
1	64.332936	34.970077	29.949277	33.448715	58.856170	35.274349
2	65.057053	34.909642	30.903789	34.909642	59.412000	36.320968
3	65.225739	37.261774	32.023096	37.261774	60.003339	39.127179
4	66.725301	39.621261	32.244810	38.042032	58.565890	39.621261
5	69.680748	39.968748	29.183551	37.563364	62.864299	40.169271

3. Implementation

As facial keypoints detection has been a popular topic of computer vision, multiple

methods are developed including active appearance models[2,3] , constrained local models[4,5], regression based methods[6] and deep-learning approaches[7−10]. Here we will focus on deep architectures since it has achieved impressive performance.

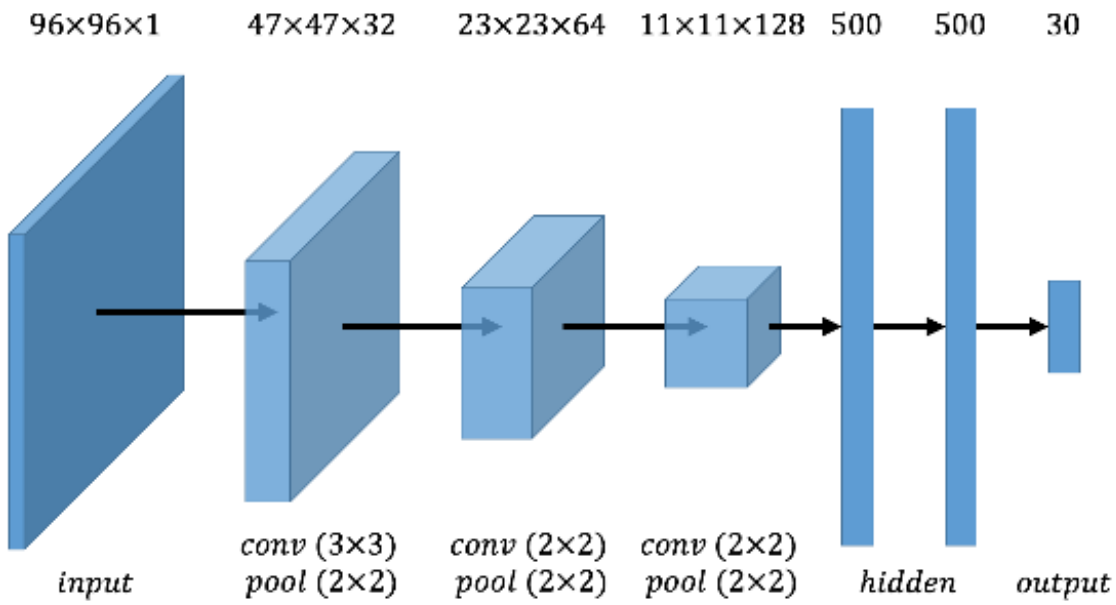
We can find that for a lot of training samples, the keypoints are incomplete. To handle the samples with missing values, we develop 2 methods:

- Method 1: Just remove them, we can refine the training set to 2140 samples and just use those with all 15 targets for training. (See model 1-7, set as label "DropNA")
- Method 2: Treat the features separately then combine them together, in this way we can fully utilise the data with missing values. (See model 8-13)

We will test both of the two preprocessing method in the project. Additionally, as larger dataset will help avoid overfitting if a model is too complex, data augmentation is then achieved by flipping images in horizontal direction.

Then different models will be implemented based on neural network with only one hidden layer and convoluntional neural nerwork. Related python modules are Keras[11] and TensorFlow[12]:

- We will first build a network with only one hidden layer, and a CNN model. These two models will be treated as baseline and we will test the influence of structure, data augmentation and number of iterations. The architecture of CNN model is:



- The traditional optimizer, SGD, is hard to choose suitable learning rate and prone to stuck in local convergence. To make some improvement, we may test some other optimizers including Adagrad[13], Adam[14] and NAdam.
- Some pretrained models can be used to extract features, which can reduce the computation time consumption and make prediction more accurate. In one model we will utilise Inception model[15] to get the features from raw image, and then treat them as inputs of our network to make prediction.
- In order to make full use of training set, we will separate the features in 6 groups, and treat each of them with same CNN model. Then these models will be combined to make prediction.

Following are our models:

Index	Structure	Preprocessing Method	Data Aug	Optimizer	Iters
1	One Hidden Layer	DropNA	N	SGD	400
2	CNN	DropNA	N	SGD	1000
3	CNN	DropNA	Y	SGD	1000
4	CNN	DropNA	Y	AdaGrad	1000

Index	Structure	Preprocessing Method	Data Aug	Optimizer	Iters
5	CNN	DropNA	Y	Adam	1000
6	CNN	DropNA	Y	NAdam	1000
7	Inception Pretrained	DropNA	Y	Adam	1000
8	CNN	left_eye_center, right_eye_center	Y	SGD	400
9	CNN	nose_tip	Y	SGD	400
10	CNN	mouth_left_corner, mouth_right_corner, mouth_center_top_lip	Y	SGD	400
11	CNN	mouth_center_bottom_lip	Y	SGD	400
12	CNN	left_eye_inner_corner, right_eye_inner_corner, left_eye_outer_corner, right_eye_outer_corner	Y	SGD	400
13	CNN	left_eyebrow_inner_end, right_eyebrow_inner_end, left_eyebrow_outer_end, right_eyebrow_outer_end	Y	SGD	400

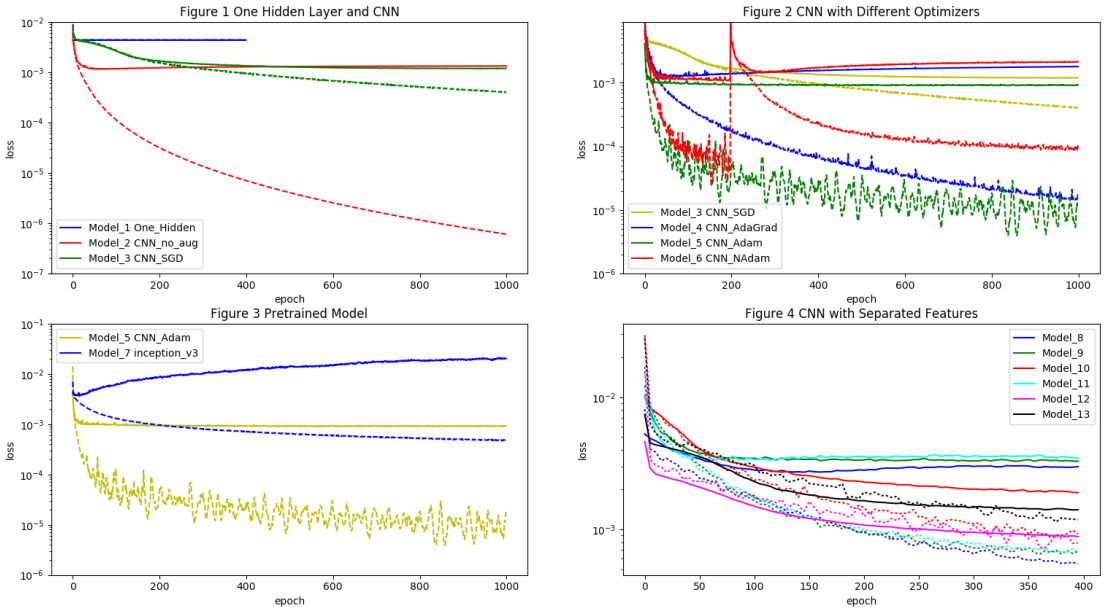
4. Result and Analysis

The result of our models is shown in the table below, MSE and Accuracy are used in training and CV part. If we submit the prediction, we can get private and public score (RMSE) from Kaggle. Note that Model 14 is just the combination of Model 8-13 to show the result.

Index	Train Loss	Train Acc	CV Loss	CV Acc	Private Score	Public Score
1	4.420e-03	0.7074	4.408e-03	0.6963	4.284	4.328
2	6.013e-07	0.9965	1.343e-03	0.8084	3.227	3.346

Index	Train Loss	Train Acc	CV Loss	CV Acc	Private Score	Public Score
3	4.065e-04	0.8560	1.214e-03	0.7897	3.250	3.319
4	1.544e-05	0.9869	1.668e-03	0.7547	3.521	3.520
5	1.778e-05	0.9756	9.272e-04	0.8037	3.036	3.125
6	8.911e-05	0.9512	2.142e-03	0.7430	3.669	3.664
7	4.820e-04	0.8709	0.02030	0.6729	8.432	8.809
8	5.360e-03	0.9978	3.033e-03	0.9943	-	-
9	6.610e-04	0.9911	3.334e-03	0.9816	-	-
10	9.230e-04	0.7572	1.910e-03	0.7611	-	-
11	6.620e-04	1.000	3.490e-03	0.9993	-	-
12	7.660e-04	0.9994	8.860e-04	1.000	-	-
13	1.094e-03	1.000	1.414e-03	1.000	-	-
14	-	-	-	-	2.027	2.449

Below are curves of decreasing loss, dashed curves represent training loss while solid curves are validation loss.



- From Figure 1 We can see that CNN model gets better score than model with only one hidden layer. In addition, the performance can be improved further by using data augmentation.
- In Figure 2, we can find that model with SGD is slow to convergence. AdaGrad and NAdam give faster result but the CV loss is high. Adam achieves best performance among all 4 optimizers that it reaches convergence fastly and the loss is the lowest.
- In our expectation this should achieve better performance, however Model 7 does not perform well. We can see from Figure 3 that this model suffers from severe overfitting and the cv loss is increasing with the epoches.
- By separating the dataset by features, we can make full use of the dataset. Compared with those use only 2140 samples, the combination of Model 8-13 achieves the best performance even with only 400 iterations that the RMSE score is much smaller than all other models

5. Conclusion and Outlook

Our best model gets 2.027 and 2.449, which ranks about top 20% on the leaderboard. By making full use of the training data, it shows better performance than those with

changed parameters only. Go beyond this project, we can make further research on following aspects if time is available:

- Try to combine separated features with optimized parameters. E.G. Separated features + Adam + more iterations
- Find the reason why pretrained model does not preform well, try to improve its performance, and test other models such as ResNet and VGG16
- Try to learn the method and experience from others.

6. Reference

[1] https://www.kaggle.com/c/facial-keypoints-detection
[2] Cootes T F, Edwards G J, Taylor C J. Active appearance models[C]//European conference on computer vision. Springer Berlin Heidelberg, 1998: 484-498.
[3] Matthews I, Baker S. Active appearance models revisited[J]. International journal of computer vision, 2004, 60(2): 135-164.
[4] Cristinacce D, Cootes T. Automatic feature localisation with constrained local models[J]. Pattern Recognition, 2008, 41(10): 3054-3067.
[5] Cristinacce D, Cootes T F. Feature Detection and Tracking with Constrained Local Models[C]//BMVC. 2006, 1(2): 3.
[6] Cao X, Wei Y, Wen F, et al. Face alignment by explicit shape regression[J]. International Journal of Computer Vision, 2014, 107(2): 177-190.
[7] Sun Y, Wang X, Tang X. Deep convolutional network cascade for facial point detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2013: 3476-3483.
[8] Liang Z, Ding S, Lin L. Unconstrained Facial Landmark Localization with Backbone-Branches Fully-Convolutional Networks[J]. arXiv preprint arXiv:1507.03409, 2015.
[9] Peng X, Feris R S, Wang X, et al. A Recurrent Encoder-Decoder Network for Sequential Face Alignment[C]//European Conference on Computer Vision. Springer International Publishing, 2016: 38-56.
[10] Fan H, Zhou E. Approaching human level facial landmark localization by deep learning[J]. Image and Vision Computing, 2016, 47: 27-35.
[11] Chollet F. Keras[J]. 2015.

[12] Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems[J]. arXiv preprint arXiv:1603.04467, 2016.
[13] Zeiler M D. ADADELTA: an adaptive learning rate method[J]. arXiv preprint arXiv:1212.5701, 2012.
[14] Kingma D, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.
[15] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 2818-2826.