Prediction of Keypoints Positions on Face Images

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper describes an approach to predicting keypoint positions on greyscale images of faces, as part of the Facial Detection (2016) Kaggle competition. Facial keypoints include centers and corners of the eyes, eyebrows, nose and mouth, among other facial features. Our methodology involves four steps to producing our output predictions. The first step involves deciding the criterion for model accuracy which is RMSE, the root means square error of all keypoint coordinates. For the second stage, we use principal components analysis to reduce the dimensionality of the image and edge-detection to better identify the key components needed to perform the facial keypoint detection. The we predict the missing feature through a neural network with a sigmoid non-linearity on the output layer. At training time, we will introduce a special convolutional neural network model which looks like an inverted hourglass and compare with the baseline model. We evaluate our different model architectures, with and without data augmentation techniques based on their Root Mean Squared Error scores they produce on the test set.

1 Background and Applications

1.1 The Development of Artificial Neural Network

Artificial Intelligence(AI) is one of the current core techs which is internationally recognized. It has already applied in a variety of fileds and subjects and made an outstanding accomplishment. And now one of the core focuses of AI is how to imitate the thinking model of human so as to present the information efficiently.

In the mid of nineteenth century, McCulloch and Pitts[1] proposed the first MP model, which laid the foundation of the early development of neural computation as well as contributed to the Perceptron Theory presented by Rosenblatt[2] in 1958. And Hopfield[3][4] proposed Hopfield Networks in 1984. The latter model mentioned provided theoretical basis for the further research on Artificial Neural Network. Two years after that, a group of scientists including Rmulhart[5] raised the theory of Parallel Distributed Processing and BackPrepagation Network with non-linear transfer function, which is named Back Propagation and has been one of the Neural Networks of greatest public concern and most usage amounts so far.

1.2 The Emergence of Deep Dearning and the Superiority of CNN

As the development of neural science, humans found that brain's signals are transmitted through a complex hierarchical structure module and learnt some characteristics of law to present observed signals from edge detection, basic shapes and more complex visual shape over time, as a consequence of which, deep learning, as a new field of researches on machine learning, appeared. The concept of deep learning was advanced by Hinton, with the purpose of analyzing and learning through building and imitating human's brains.

CNN is a machine learning module from deep supervised learning with high adaptation. It's good at mining data's local characteristics, extracting global training characteristics and classifying, its structure of weighted sharing networks makes it more similar to biological neural network, and reaches a lot of accomplishment in various fields of pattern recognition

1.3 The Appliances of Facial Keypoints' Prediction

The research on facial keypoints' prediction including cross knowledge from the fields of computer vision, pattern recognition, picture processing, machine learning and computer graphics. The technology of facial keypoints' prediction can be applied in a lot of ways such as face recognition, facial expression recognition, attitude estimation, age estimation, face reconstruction, computer animation module construction, current human computer interactions, driver fatigue monitoring and facial aesthetics analysis, etc. Sharp image can enhance the accuracy of human face detection. Facial keypoints' prediction has been widely used in public security, social security and commercial field of state.

2 Review of the State-of-the-art

Prediction of key point positions on face images is a difficult and popular research topic. In recent decades, a considerable number of researchers have conducted in-depth research, and has made rich some results. At present the state-of-the-art facial key point location methods are the following:

2.1 Statistical Regression Method Based on Discriminative Learning

Xiong Proposed a Supervised Descent Method (SDM) to solve the nonlinear least squares problem and applied it to Prediction of key point positions on face images [6]. In this method, a series of gradient descent directions are learned from the training data and a corresponding regression model is established. This model is used to estimate the gradient direction in the test. It is not necessary to calculate Jacobian matrix and Hessian matrix. This method has been improved[7], which makes the discriminant model can be updated, and can construct a personalized identification model for individuals, and obtain more robust key point positions' tracking results. In[8], sparse shape constraints are introduced to the cascade regression model to reduce the uncertainty of the gradient descent direction, while the low rank representation is applied to the image to reduce the attitude effect.

Some researchers have used the classical SVM to train discriminant local feature detectors to locate local feature points[9]-[12]. In[12], feature point localization is treated as an example of the structured output classification problem, and Structured Output SVM (SO-SVM)[13] is used to supervise the learning of feature points. This method has better location effect in In-the-wild database. The Support Vector Regression (SVR) and Markov Random Fields (MRF) are combined into regression prediction in the localized region of the constraint[14]. The improved method[15] in the probability map model focus the way to estimate a robust feature point prediction position.

The method based on Random Regression Forests[16] also shows good performance in the localization of facial feature points[17]-[21]. This kind of method basically uses the leaf node of each tree in the forest to vote for the candidate area around the key point, and then gather statistic of voting result to get the key point location with the greatest support. Dantone et al. Proposed a Conditional Regression Forests (CRFS)[18] to locate real-time key points for low-quality images. For different head states, they train multiple sets of regression trees based on different head pose, and then use the conditional probability of the feature image block belongs to head posture to calculate the confidence level of the event that the feature block falling into a leaf node. In order to remove some interference polls, Yang and Patras proposed a method of screening polls, filtered out polls inconsistent with the presumption of the central position of the human face, and adjusted the classification of extracted features in real time[21] . Because the random forest method can vote for the candidate regions of each feature point, there is no burden of shape constraints, so it is more universal. But it is very sensitive to the local occlusion problem because of the lack of global shape constraints.

2.2 Method Based on Deep Learning

Many excellent machine learning models, such as SVM, Boosting method, maximum entropy method, etc., can be considered as a shallow structure algorithm. Such methods rely on hand-designed features (such as gradient features, SIFT features, Gabor features, etc.). A good feature can greatly improve the performance of machine learning and pattern recognition system. Similarly, inappropriate features will seriously reduces the performance of the system. Compared with the shallow algorithm, the deep learning, with the help of big data, constructs a machine learning model with many hidden layers to learn new and useful features from the massive training data for new applications, and finally enhance the classification or prediction accuracy. In recent years, researchers have introduced several commonly used deep learning models into the study of face key point localization technology and have achieved great success, which is one of the front methods of face feature point location.

In[22], Luo et al. use Deep Belief Network (DBN) to train face segmentation model, and use Logistic Regression to discriminate adjustment. The hierarchical analysis using deep learning provides rich facial feature information, which can accurately gain face key points location and facial features segmentation. In[23], the Deep Convolutional Neural Networks (CNN) are used to locate the key point positions on face images from rough to precise. In [24], Zhang et al. used Stacked Auto-encoder Networks (SAN) for model training. In[25], a prior model of face shape was constructed using Restricted Boltzmann Machines (RBM), which captures the information of positive and negative changes caused by facial gesture and facial expression, then we can locate key point in any expression or posture. Deep learning model use the multi-layer nonlinear mapping to separate facial expression, age, light and other factors, changing the various factors into a simple linear relationship. They no longer interfere with each other, which can help effectively solve all kinds of people face analysis problem. Compared with the traditional methods, however, the deep learning's training methods require a lot of experience and skills.

3 Introduction

3.1 Problem Summary

The purpose of this paper is to predict the (x, y) real-valued pair of 15 facial keypoints for a given set of 96-by-96 grayscale images.

There are 15 keypoints:

- Left and right eye center (2)
- Left and right eye inner and outer corners (4)
- Left and right eyebrow inner and outer ends (4)
- Nose tip (1)
- Mouth left and right corner (2)
- Mouth top and bottom lip center (2)

In 1, a sample of images and the associated keypoints are shown for reference. Note that only two have the full set of keypoints plotted, most faces are missing at least multiple keypoints. And this is also a feature of the data. Accordingly, we predict two values for each keypoint: (1) whether it is present in an image or not, and (2) what its (x, y) coordinates would be if it exist.

As required, we randomly subset the data so that 90 percent of images are for training a model and the other 10 percent are for validation.

The criterion for model accuracy is the root means square error of all keypoint coordinates:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Figure 1: Random Sub-Sample of Faces and Keypoints

3.2 Data

Because most images don't contain all keypoints, there's a problem when we are trying to train our model. One naive solution would be to throw out all faces without all keypoints, but doing so would remove 4909 images from our training data set (out of 7049 total images) – about 70 percent of our data.

An important observation is that keypoints tend to be missing in groups, so that (as an example) if an image does not have one of the eyebrow keypoints, it tends to also not have all other eyebrow keypoints. Accordingly, we bundle the keypoints into the following groups (the number of images containing all keypoints in the group are reported in parentheses):

- eye center: left eye center, right eye center (7033)
- eye corner: left eye inner corner, left eye outer corner, right eye inner corner, right eye outer corner (2247)
- eyebrow: left eyebrow inner end, left eyebrow outer end, right eyebrow inner end, right eyebrow outer end (2190)
- mouth (bottom): mouth center bottom lip (7016)
- mouth (excl. bottom): mouth left corner, mouth right corner, mouth center top lip (2260)
- nose: nose tip (7049)

Then we can group each feature with other features that tend to be present in the same image. Within a group, we throw out images in which all keypoints *in that group* are not present, but in 22 out of 30 cases this requires us to remove less than 1 percent of the available images (in the other 8 cases, we throw out between 1 and 4 percent of available images).

To divide the features into groups like this, we should train 6 models, but each model will have more available training data than a single model required to predict all features.We will also need to predict whether or not a keypoint is contained in a feature, so that we can predict an empty value for keypoints that are likely to be missing in the test set. We discuss this model more fully in 4.

3.3 Image Pre-processing

A common practice in other computer vision studies is to use principal components analysis to reduce the dimensionality of the images. In most cases, nearly all the variation across images can be explained with a fraction of the original data set. Rather than defining each image as a vector of length $p = 96 \cdot 96$, we can express it as a linear combination of the first x principal components where $x \ll p$. This process should reduce the noise in each vectorized image and increase the accuracy of the predictive algorithms we need to train and tune.

In addition to PCA, we will also employ edge-detection to help us better identify the key components needed to perform the facial keypoint detection. Some of the filtering kernels we have already employed on the data-set can be seen in

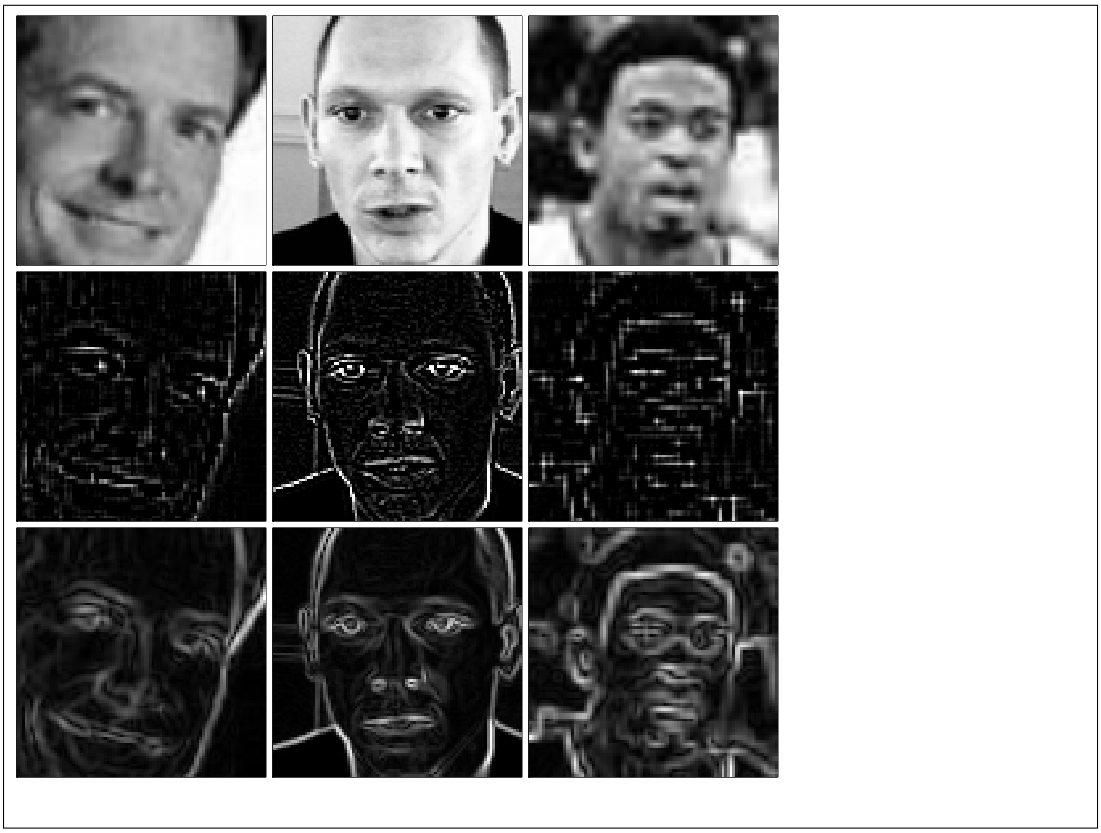


Figure 2: Sample figure caption.

(see the second and third lines). We may end up combining this with neural network to create a multi-stage convolutional neural network. We may explore the techniques used in the LeNet-5 by Yann LeCun as he has a tutorial on how to use Python to do something similar to what we're trying to do.

3.4 Tools and Software

The main tool we use in this paper is a multilayer convolutional neural network. We built and trained the model in Python using the Theano and Lasagne libraries.[1]

[1]Lasagne is "a lightweight library to build and train neural networks in Theano." See `https://github.com/Lasagne/Lasagne` for details.

4 Missing Feature Model

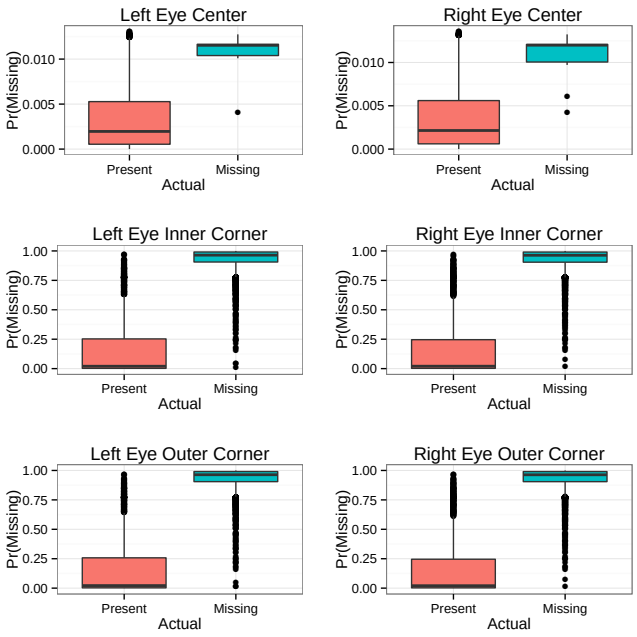
Because the data contain many missing features, our resulting model must also predict the presence of features. To predict this we turn to a slightly modified version of the neural network described in 5 with a sigmoid non-linearity on the output layer and the binary cross-entropy loss function defined below:

$$L = -target \log(p) - (1 - target) \log(1 - p)$$

where `target` indicates whether the feature is actually missing ($\in \{0, 1\}$) and `p` is the predicted probability of the feature being missing

For many of the features, the neural network does a good job separating the two classes and producing probabilities that make logical sense. The ones that it doesn't separate well, like `Left Eye Center`, `Right Eye Center` and `Mouth Center Bottom Lip`, have few missing data-point exemplars in the training data (9, 9, and 23 respectively out of 4934 samples).

Figure 3: Boxplots for Missing Eye Features



With the output trained, we next turn to determining the optimal cutoff. Using the R package `OptimalCutpoints` we choose to maximize the accuracy area [, lewis2008use,greiner1995two,greiner1996two] which is defined in 4.

$$AA(c) = \frac{TP(c)TN(c)}{(TP(c) + FN(c))(FP(c) + TN(c))}$$

where TP = True Positives, TN = True Negatives, FN = False Negatives, and FP = False Positives

As the boxplots show, there are generally two categories of features: those that are missing often and those that are not. For the ones that are missing often, we have a relatively good separability between the predicted probabilities. For those that are not missing often, our predictive power diminishes. Turning to the ROC plot for a feature that is missing quite often, `Left Eye Inner Corner`, (see 6) we see there is a lot of area under the ROC curve and the optimal cutoff point of 0.5 seems to put us right on the upper-left edge of the curve. Next, looking at `Left Eye Center`, a feature with few missing exemplars, (see 7) we see less area and a clear trade-off between specificity and sensitivity. When the value is 0.5, it seems to put us on the upper-left edge of the curve.

Figure 4: Boxplots for Missing Eyebrow Features

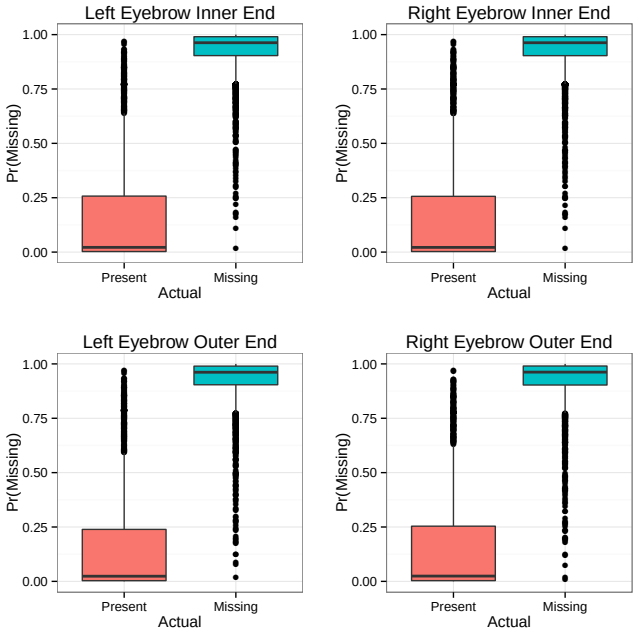
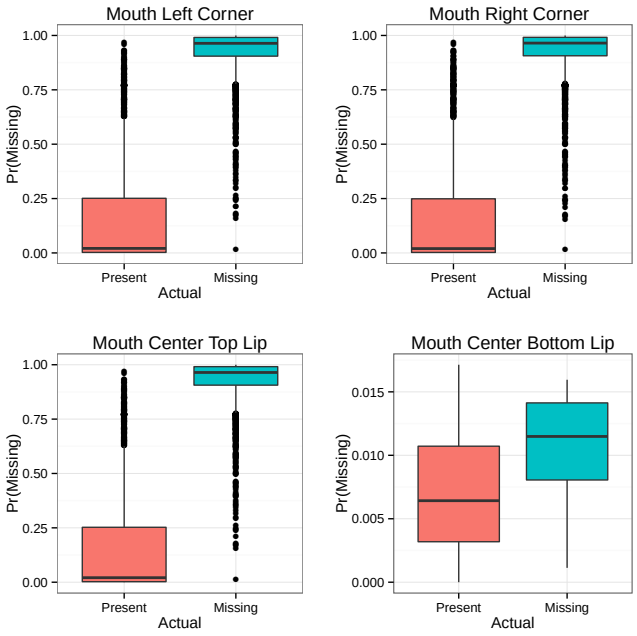


Figure 5: Boxplots for Missing Mouth Features



So given the optimization condition, the best cutoff for all of these features is 0.5 (see 1[2]). In each of the cases, we end up with quite a large sensitivity and a reasonable specificity.

[2]Note: that the `Nose Tip` feature is present in all training and validation images.

Figure 6: ROC Curve for Left Eye Inner Corner

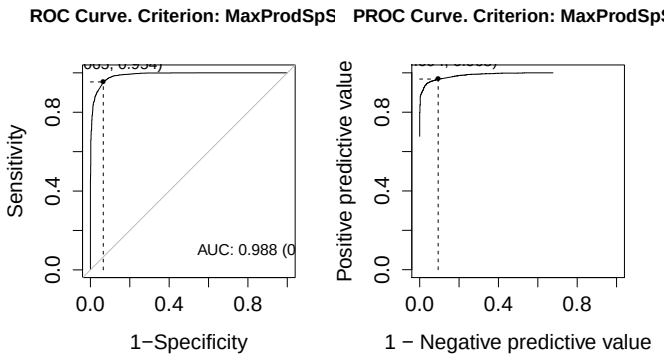
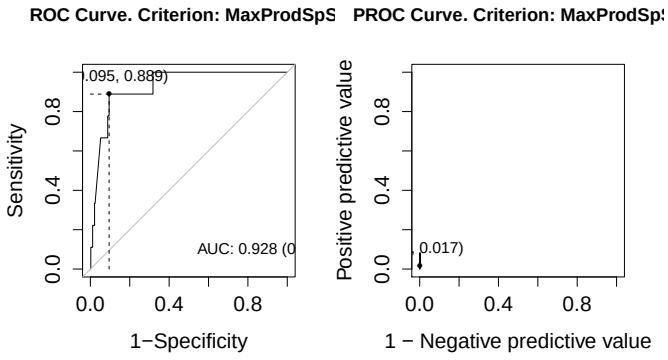


Figure 7: ROC Curve For Left Eye Center Feature



5 Individual Keypoint Models

Below we will introduce a subset of the models we experimented with and report performance statistics for each. We will also discuss some auxiliary modeling techniques that we investigated. For each of the models we discuss below, we use a rectifier nonlinear activation and adjust the learning rate and momentum dynamically.

Table 1: Cutoff Validation Performance

Feature	Cutoff	Sensitivity	Specificity
Left Eye Center	0.500	0.000	1.000
Right Eye Center	0.500	0.000	1.000
Left Eye Inner Corner	0.500	0.961	0.714
Right Eye Inner Corner	0.500	0.962	0.715
Left Eye Outer Corner	0.500	0.960	0.713
Right Eye Outer Corner	0.500	0.961	0.715
Left Eyebrow Inner End	0.500	0.961	0.714
Right Eyebrow Inner End	0.500	0.961	0.716
Left Eyebrow Outer End	0.500	0.963	0.713
Right Eyebrow Outer End	0.500	0.963	0.720
Nose Tip	0.500		
Mouth Left Corner	0.500	0.961	0.718
Mouth Right Corner	0.500	0.960	0.716
Mouth Center Top Lip	0.500	0.960	0.717
Mouth Center Bottom Lip	0.500	0.000	1.000

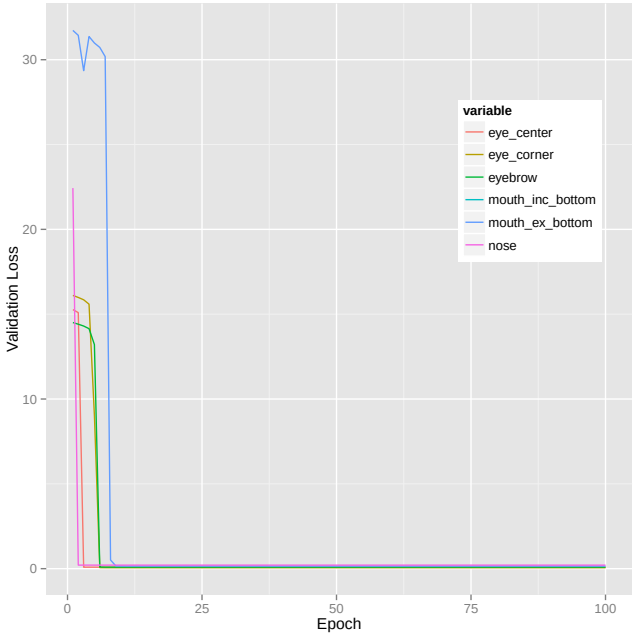
5.1 Baseline Model

The baseline model is a convolutional neural network, which has six layers: 3 convolutional/pooling layers, and 3 hidden layers. We began our modeling with only three layers, but following [, benlecun2007], our idea was that by increasing the depth of the network, we would increase its predictive accuracy as well. In theory, additional layers should allow the network to discover or learn more than just the low-level features of the images. And we hope that it would increase its ability to identify specific keypoints.

We also hope that convolutional layer would allow the network to automatically train 2-dimensional filters. The hope is that the neural network will automatically adapt the filter taps to detect low-level features like edges. Higher-level features'(like eyes', noses', mouths')training is also possible.

The loss function on the validation data set for this set of models is displayed in 8. The RMSE on the validation set is 3.85 (see 2).

Figure 8: Validation Loss for Feature Detection Models (Baseline)

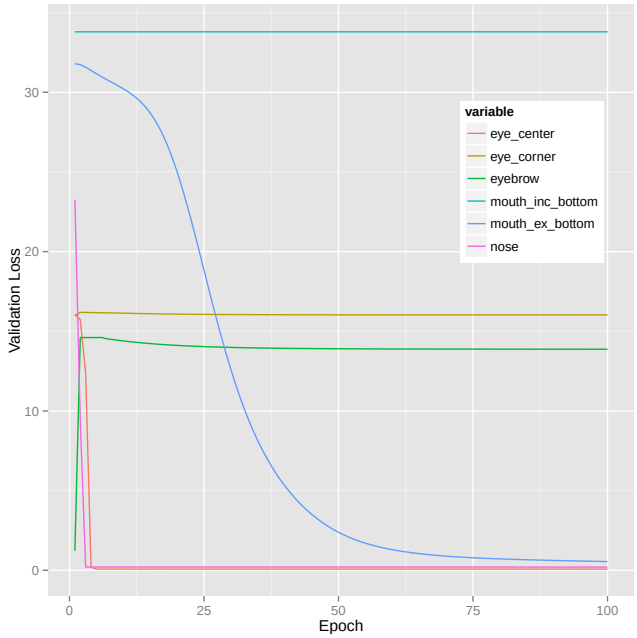


5.2 Baseline Model with L1 and L2 Regularization

To strengthen our baseline model ,we used L1 and L2 regularization. Compared to the model without regularization, we find that the loss function falls more gradually and more smoothly. This is because the cost function now has a second term that penalizes model complexity, so that early on in the training cycle the model is less able to fit the data. By not over-fitting the data, we hope that the model is better at understanding images that it has not seen before. In other words, we are hoping to move the model more towards the optimal trade-off between bias and variance, and has a better generalization ability.

The loss function on the validation data set for this set of models is displayed in 9. The RMSE on the validation set is in excess of 35 (see 2), making it far and away our worst performing model. One explanation is that the model has been over-regularized; we have put too much of a premium on building a simple model, so that the result is very bad predictive capability. We can see this a bit in 9, as the loss function for three of the feature models are not able to stabilize even after 100 epochs.[3]

Figure 9: Validation Loss for Feature Detection Models (Baseline with Regularization)



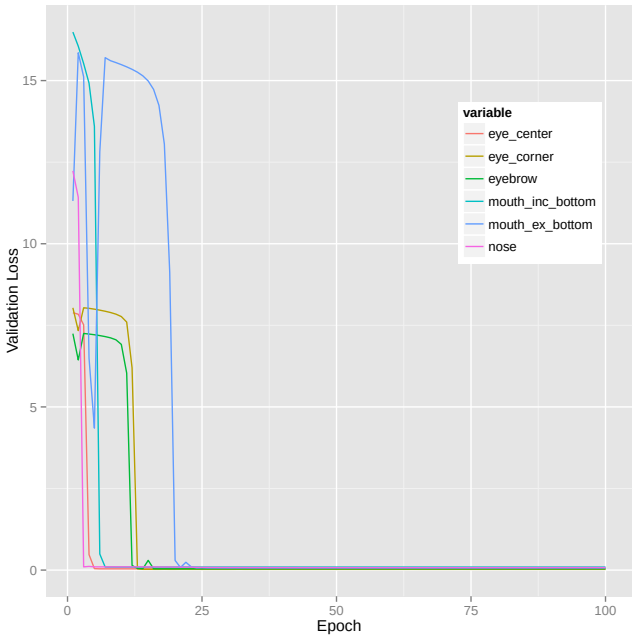
5.3 Inverted Hourglass

As we explored different neural network structures, we coined the term inverted hourglass. In this model we decided to place more convolutional filters near the input to see if it would improve over our existing model. Therefore, we essentially flipped the convolutional part of our initial model over (hence the inverted part of the name) to start with 32 filters at the beginning and whittling this down to eight just before the densely connected layer. Having a much smaller set of filters at the output of the convolutional layer meant that we would use a smaller initial dense layer, which we then doubled and halved again before the output layer (hence the hourglass part).

The loss function on the validation data set for this set of models is displayed in 10. The RMSE on the validation set is 3.70, making it our best performing model (see 2).

[3]We show the results here for 100 epochs; we trained some models for up to 1000 epochs but saw no performance improvements.

Figure 10: Validation Loss for Feature Detection Models (Inverted Hourglass)



5.4 Inverted Hourglass with Regularization

Now we strengthen the 6 layer model by including only the L1 regularization. Once again, the goal here is to prevent the model from over-fitting to the images that it trains on, so penalizing model complexity should increase the out-of-sample accuracy. Following [, ng2004feature], we chose L1 regularization because we have a dimensionally large input data set (each image has 96x96 pixels, with each pixel being an input feature).

The loss function on the validation data set for this set of models is displayed in 11. The RMSE on the validation set is 3.98; predictive performance has declined by adding regularization.

5.5 Inverted Hourglass with Regularization and Dropout

At last, we strengthen the 6 layer model with L1 regularization by including dropout. Similar to regularization, drop helps us ensure that we're not focusing too much on the training set, trying to find the real underlying structure. Again, the goal is to choose a simpler model, but instead of adjusting the cost function we instead alter the structure of the network by deleting nodes in the intermediate layers during training.

The loss function on the validation data set for this set of models is displayed in 12. The RMSE on the validation set is 4.40; predictive performance has declined again by combining dropout with regularization.

Table 2: RMSE for Different Model Structures

Model	RMSE
Inverted Hourglass	3.70
Inverted Hourglass with Regularization	3.98
Inverted Hourglass with Regularization, Dropout	4.40
Baseline	3.85
Baseline with Regularization	35.21

Figure 11: Validation Loss for Feature Detection Models (Inverted Hourglass with Regularization)

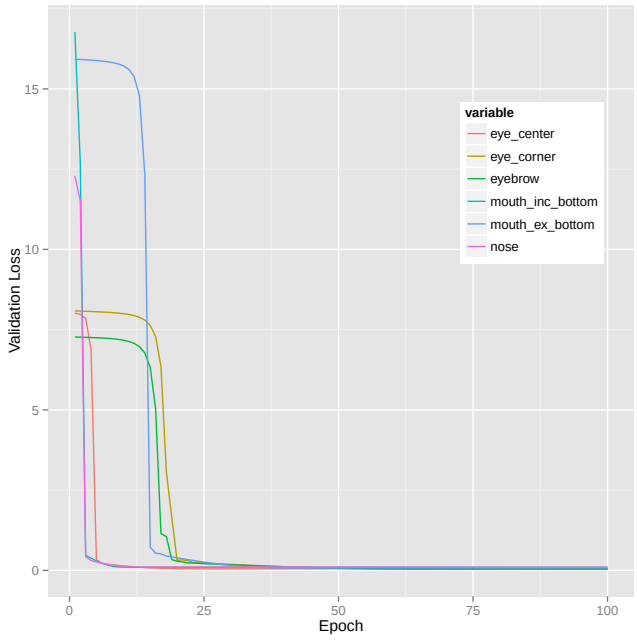
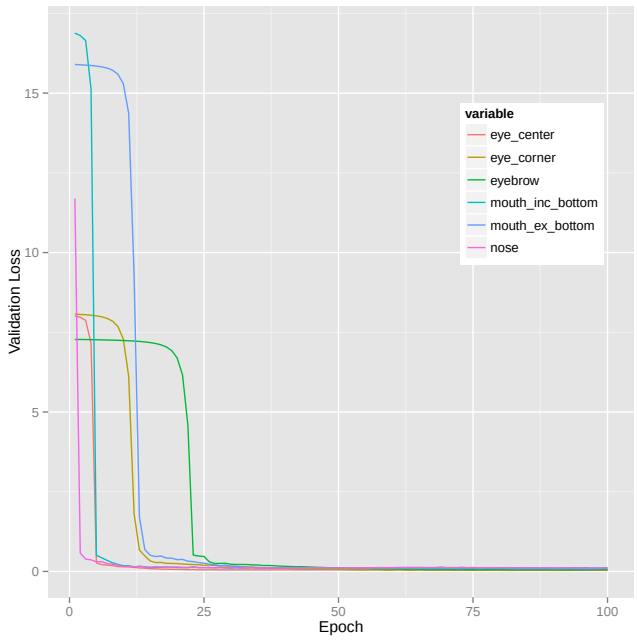


Figure 12: Validation Loss for Feature Detection Models (Six Layers)



5.6 Other Modeling Choices

But we are still not satisfied with this result, so we tried many other methods to see if we could train the model a bit better. These experiments will be discussed below.

5.6.1 Batch-Size

Although the batch-size should not matter, we did want to see if it had any effect on the training. As it turns out, we did notices that larger batch-sizes did help lower the resulting RMSE slightly. This was likely because the nesterov momentum update algorithm could incorporate a larger amount of the data-set in each update.

5.6.2 Learning-Rate

At the beginning, we set the learning rate too high, and quickly saturated our 32-bit floating-point accumulators. Additionally, we knew that if we set it too low, we would wait forever to converge to an answer and, worse-yet, risked ending up in a local minima.

So we should acheive the balance. After some consideration we saw that dynamic learning-rates were common-place [, gorr,dnouri] and we opted to implement and experiment with both a log and a linear learning-rate backoff based, for simplicity, solely on the number of epochs trained.

5.6.3 Momentum

Our update algorithm (nesterov updates) included both a learning-rate and momentum parameter. After reading a bit in [, gorr] about the advantages of momentum in the overall amount of time needed to train a network we set about determining a reasonable value. Further reading in the lasagne manual [, lasagnenesterov] led us to experiment not only with static momentum parameters but also with dynamic ones that increased momentum as learning-rate decreased.

5.6.4 Dropout

We found that too little and too much dropout had negative effects on the training. In the former case, we likely were over-fitting our data, in the latter one, we introduced too much noise to train on. Finally we found the appropriate value of dropout is 10-20%.

5.6.5 Scaling Output

At last, we tried to see if we could get the neural network to train a little better. We scaled the output to lie between -1 and 1 in one experiment. In another we tried normalizing it to so that coordinate positions were measured in standard deviations away from the mean. And we found that reported training RMSEs were higher since the scale was smaller, but the resulting de-scaled RMSEs were worse. So, unfortunately, we did not see improvements in the RMSEs produced by either of these experiments.

5.7 Image Rotating

According to our models outputs, there is a key observation: most of the predicted keypoints are very close to the average location of each keypoint. That is, to a first approximation our models just predict a keypoint will be at its average location; the variance is possibly due to the model learning to understand localized variation in the images. One idea to help the model learn the ”true” features of each face would be to pre-process the images by rotating them. Our goal would be that the model could not simply predict average locations for each keypoint, but would have to truly learn what signals the location of the facial features.

Thus we augmented the training data set with 3 copies of each image, rotated 90, 180, or 270 degrees. It appeared as though the model was still, to a first approximation, predicting the average location of each keypoint, although because the images were rotated, the average location was nowhere close to the actual location for any of the images. Unfortunately, the results did not meet our expectations, as the RMSE of the model trained on the augmented data set exceeded the RMSE of the baseline model by at least 2 times.

5.8 Learning Representation

In addition, we also used the penultimate layer of our baseline neural network as the input into a different type of model (random forest or penalized regression, in our case). The idea is to use the neural network to learn and identify important features of the data, and then to use these important features in a final predictive model. Unfortunately, this methodology did not produce successful results in our case. We did not investigate it any further becaue the RMSE produced by this additional layer of modeling was worse than the traditional baseline model.

5.9 Combined Model

We tried to train both the missing data probability and the feature coordinate locations simultaneously in the combined model. This involved a fair amount of additional work to better understand the inner-workings of Theano and Lasagne, but the idea was that if we could train both models at once, we wouldn't have to train 7 models in total (6 feature-sets + the missing probability). Additionally, because the neural network had to detect the presence of the missing feature, it was hoped it could use this to predict location better. Finally because the neural network could see all images in the training dataset, it could theoretically do a better job learning. To do this, we had to create a loss-function that combined RMSEs from the keypoint coordinates and Binary-Cross-Entropy values from the logistic output. To make matters more complicated, we had to write this loss-function to not penalize the network for a coordinate given on a missing feature. Unfortunately, when the model was built and run, it fared worse than the model trained on the 7 separate features. It is likely that a better loss-function could have helped this, but because of the time we have to give up.

6 Discussion

6.1 Best and Worst Performances

We show an average face which is a composite of all images in our data set in 13. On the image is the average location of each keypoint (marked with a "+" sign), and the shaded circles show the average deviation of the predicted keypoints from the actual keypoints. Just like eye corners were all predicted with one model so that they are all depicted in yellow, colors correspond to the model used for each set of keypoints.

There are two different ways to judge the accuracy of our model: the RMSE of each keypoint coordinate (shown in 3), or the Euclidean distance of the predicted keypoint (x, y) coordinate from the actual (x, y) coordinate (shown in 3; these are represented by the circles in 13).

According to 4 which clearly lists the distances between predicted in actual keypoints, we can see that the model is best at predicting eye features. The top of the list (ordered from most to least accurate) is comprised entirely of the eye-related features (eye centers, eye corners, and eyebrows). We can see in 13 that the radii around the eye features are smaller than those of the nose and mouth features. The most accurate keypoint is the inner corner of the right eye, which has a distance between the predicted and actual keypoints of just over 3 pixels.

On the contrary, it is not very good at predicting where the tip of a nose is. It is the least accurate keypoint, with a mean distance between the predicted and actual value of over 6 pixels.

Table 3: RMSE for Predicted Keypoints

Keypoint	RMSE
right eye inner corner x	2.46
mouth center top lip x	2.48
right eye inner corner y	2.57
right eyebrow inner end y	2.68
left eye inner corner y	2.70
right eyebrow inner end x	2.71
left eyebrow inner end y	2.72
right eye outer corner y	3.04
left eyebrow inner end x	3.05
left eye outer corner y	3.07
right eye outer corner x	3.17
mouth left corner x	3.26
left eyebrow outer end y	3.29
right eye center x	3.37
right eyebrow outer end y	3.38
right eyebrow outer end x	3.39
mouth right corner x	3.42
left eye inner corner x	3.48
right eye center y	3.50
left eye center y	3.57
left eyebrow outer end x	3.59
left eye center x	3.75
left eye outer corner x	4.01
mouth right corner y	4.29
mouth center bottom lip x	4.32
nose tip x	4.34
mouth left corner y	4.62
mouth center top lip y	5.22
mouth center bottom lip y	5.60
nose tip y	5.91

Table 4: Average Euclidean Distance between Predicted and Actual Keypoints

Keypoint	Average Distance
right eye inner corner	3.17
right eyebrow inner end	3.32
left eyebrow inner end	3.55
right eye center	3.57
right eye outer corner	3.89
left eye inner corner	3.96
left eye center	3.97
right eyebrow outer end	4.10
left eyebrow outer end	4.18
left eye outer corner	4.43
mouth right corner	4.74
mouth left corner	4.82
mouth center top lip	4.91
mouth center bottom lip	5.82
nose tip	6.07

6.2 Model Evaluation

A criterion for judging model performance is how well we predict relative to naive averaging of keypoints. That is, one simple algorithm would be to predict–for all faces in the data set–the average of each keypoint across all faces. Doing so yields an average RMSE (across all keypoints) of 3.6.

Figure 13: Average Face with Estimated Keypoints

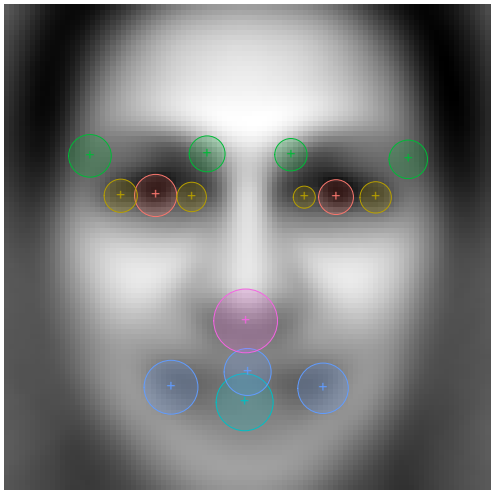
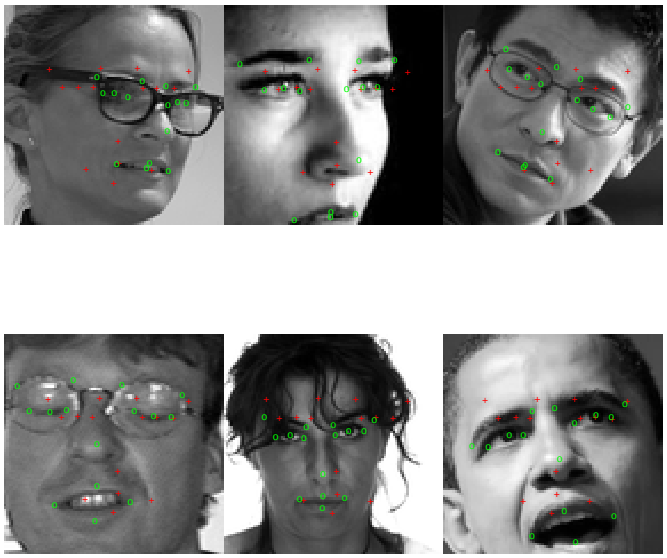


Figure 14: Most Accurate Predictions (Average RMSE = 1.768764)

This actually compares favorably to the predictive capability of our best model, which produces an RMSE of 3.7. It appears that our model with this configuration can not really learn the features of facial keypoints.

When we examined the badly mis-predicted faces in 15, one feature stands out – it seems that the model tends to predict keypoints close to their average values regardless of how the face in a given image is oriented. We can see this playing out in the plots above showing the validation losses for each model; in most cases, the validation loss relatively quickly drops then stays constant for the

Figure 15: Least Accurate Predictions (Average RMSE = 7.874467)



remainder of the training run. One interpretation of this is that the model quickly learns where the average keypoints are on an image, then fails to learn any other features of the data sets. It is possible that there are too few exemplars with tilted faces for the model to train against, so it could not handle these features well.

Reference

[1]McCulloch W S, Pitts W. A logical calculus of the ideas immanent in nervous activity[J]. The bulletin of mathematical biophysics, 1943, 5(4): 115-133.

[2]Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain[J]. Psychological review, 1958, 65(6): 386.

[3]Hopfield J J. Neurons with graded response have collective computational properties like those of two-state neurons[J]. Proceedings of the national academy of sciences, 1984, 81(10): 3088-3092.

[4]Hopfield J J, Tank D W. Computing with neural circuits- A model[J]. Science, 1986, 233(4764): 625-633.

[5]Rumelhart D E, Hinton G E, Williams R J. Learning Internal Representations by Error Propagation, Parallel Distributed Processing, Explorations in the Microstructure of Cognition, ed.DE Rumelhart and J. McClelland. Vol. 1. 1986[J]. 1986

[6]X. Xiong, F. De la Torre. Supervised Descent Method and its Applications to Face Alignment [C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2013:532-539.

[7]A. Asthana, S. Zafeiriou, S. Cheng, et al. Incremental Face Alignment in the Wild[C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2014:1859-1866.

[8]J. Deng, Y. Sun, Q. Liu, et al. Low Rank Driven Robust Facial Landmark Regression[J]. Neurocomputing, 2015, 151:196-206.

[9]C.-T. Liao, Y.-K. Wu, S.-H. Lai. Locating Facial Feature Points Using Support Vector Machines[C]. Proceedings of IEEE International Workshop on Cellular Neural Networks and Their Applications. IEEE, 2005:296-299.

[10]V. Rapp, T. Senechal, K. Bailly, et al. Multiple Kernel Learning Svm and Statistical Validation for Facial Landmark Detection[C]. Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (FG). IEEE, 2011:265-271.

[11]C. Du, Q. Wu, J. Yang, et al. Svm Based Asm for Facial Landmarks Location[C]. Proceedings of IEEE International Conference on Computer and Information Technology. IEEE, 2008:321-326.

[12]P. N. Belhumeur, D. W. Jacobs, D. Kriegman, et al. Localizing Parts of Faces Using a Consensus of Exemplars[C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2011:545-552.

[13]M. Uricar, V. Frac, V. Hlavac. Detector of Facial Landmarks Learned by the Structured Output Svm[C]. Proceedings of International Conference on Computer Vision Theory and Applications (VISAPP). Springer, 2012:547-556.

[14]I. Tsochantaridis, T. Joachims, T. Hofmann, et al. Large Margin Methods for Structured and Interdependent Output Variables[J]. Journal of Machine Learning Research, 2005, 6(2005): 1453-1484.

[15]M. Valstar, B. Martinez, X. Binefa, et al. Facial Point Detection Using Boosted Regression and Graph Models [C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2010:2729-2736.

[16]B. Martinez, M. F. Valstar, X. Binefa,et al. Local Evidence Aggregation for Regression-based Facial Point Detection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(5):1149-1163.

[17]N. Duffy, D. Helmbold. Boosting Methods for Regression[J]. Machine Learning, 2002, 47(2-3):153-200.

[18]G. Fanelli, J. Gall, L, Van Gool. Real Time Head Pose Estimation with Random Regression Forests [C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2011:617-624.

[19]M. Dantone, J. Gall, G. Fanelli, et al. Real-time Facial Feature Detection Using Conditional Regression Forests[C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2012:2578-2585.

[20]G. Fanelli, M. Dantone, J. Gall, et al. Random Forests for Real Time 3d Face Analysis[J]. International Journal of Computer Vision, 2013, 101(3):437-458.

[21]G. Fanelli, M. Dantone, L. Van Gool. Real Time 3d Face Alignment with Random Forests-based Active Appearance Models [C], Proceedings of IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG). IEEE,2013:1-8.

[22]H. Yang, I. Patras. Sieving Regression Forest Votes for Facial Feature Detection in the Wild [C]. Proceedings of IEEE International Conference On Computer Vision (ICCV). IEEE, 2013:1936-1943.

[23]P. Luo, X. Wang, X. Tang. Hierarchical Face Parsing via Deep Learning[C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPP). IEEE, 20l2:2480-2487.

[24]Y. Sun, X. Wang, X. Tang. Deep Convolutional Network Cascade for Facial Point Detection[C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2013:3476-3483.

[25]J. Zhang, S. Shan, M. Kan, et al. Coarse-to-fine Auto-encoder Networks (CFAN) for Real-time Face Alignment [C]. Proceedings of European Conference on Computer Vision (ECCV). Springer, 2014:1-16.