Attention Map Comparison between Support Vector Machines and Convolutional Neural Networks

Lei Chen
New York University
lc3909@nyu.edu

Yusen Su
New York University
ys3547@nyu.edu

Fangjun Zhang
New York University
fz758@nyu.edu

Abstract

Image classification, particularly face-related classification task, is one of the keystones for machine vision. Facial expression is a complicated behavior of human beings and its recognition is challenging for computers. In this project, we evaluate statistical machine learning methods such as Support Vector Machines (SVMs), Principal Component Analysis (PCA), AdaBoost and Random Forest to tackle this problem. Furthermore, with the emerging development of deep learning, we take convolutional neural networks (CNNs) into consideration as well. Apart from metric evaluation, it is also important to understand what and how the machine learns. Hence, we explore the visualization methods for linear SVM, linear PCA as well as CNNs, called activation map. We also visualize the internal representation of SVM and PCA to argue the judgment is significantly based on the pattern of mouths. We analyze the limitations of our method while applying to statistical learning methods with non-linear kernel at the end of our report.

1. Introduction

Facial expression is a complicated human behavior reflecting people's emotion. Although it is traditionally a psychological topic, facial expression recognition is also an active research area in machine learning and computer vision. Statistical machine learning methods such as Support Vector Machines (SVMs) [4], Principal Component Analysis (PCA) [17], Random Forest [11] and Adaboost [7] were invented to process generic data and also suitable for facial recognition recognition. Recently, deep learning methods such as convolutional neural networks show significant improvement in many computer vision tasks such as image classification [5], object detection and semantic segmentation. In this project, we aim at evaluating and comparing the classification performance of both statistical learning and deep learning algorithms. Furthermore, we are interested in how and what the computer learns about the dataset during its training procedure. We proposed a simple visualization method for statistical machine learning methods to show the internal representation and their activation map conditioned on input images. Our method uses learned weights as representation and Hadamard product between weights and vectorized images as activation maps. As for visualization method for CNN, we choose classification activation map (CAM) [27] to explore the response of neural network models given images. CAM utilizes the global average pooling operations and output projection, and replace dot product with Hadamard product to obtain activation map. We use the FER2013 dataset [8] for training and validation. Note that we aim at visualization and explainability of each machine learning algorithm rather than performance. In the performance comparison, we reported that CNN models have overwhelming advantages over traditional statistical learning methods. For visualization experiments, we first show the internal representation of SVM only and SVM with PCA dimension reduction. Also, we compare activation maps between SVM, SVM with PCA and ResNet [9]. We analyze our visualization results, and observed that mouth patterns are positively significant to the final classification while cheek regions are negatively impacts. We also found that usage of our visualization methods cannot be generalized in the presence of nonlinear kernel. The future directions to extend our project include i) apply kernel methods to our visualization method and ii) evaluation of partially occluded facial expression recognition.

2. Related Works

2.1. Statistical Learning

Support Vector Machine (SVM) [4] has been widely used in various pattern recognition tasks. It is believed that SVM can achieve a near optimum separation among classes [3]. One remarkable property of SVMs is that they have ability to learn independently of the dimensionality of the feature space and focus on learning raw pixels on image independently. Meanwhile, **Principal Component Anal-**

ysis (PCA) is a great dimension-reducing tool that can be utilized as combining old features to new characteristics, as well as emphasizing data variation so that summarizing patterns from features. The most important part for image visualization has been found by using PCA in preprocessing to extract feature map [25]. In the meantime, not only decreasing the size of feature matrix that increases the speed of the classification and but also using SVM as a classifier after the extracted features have good accuracy displaying the expression and facial action units[23].

In addition to SVM and PCA, Adaboost and random forest are also applied to the classification task, as classic ensemble methods. This part of work is mainly on tuning a bunch of parameters. **Adaboost** utilizes a weaker learner to approximate labels of approximation at each step then modifies weight of each example for the next step. Tuned parameters of AdaBoostClassifier include `n_estimators`, `learning_rate` and `max_depth` of `DecisionTreeClassifier` as the base estimator[18]. **Random Forest** fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control overfitting. Tuned parameters of RandomForestClassifier include `max_depth` and `n_estimators`[18].

2.2. Convolutional Neural Networks

Deep learning algorithms for image related tasks are build up by CNNs. The first foundations work is LeNet [13] for hand-written digits classification. With more large-scale image classification datasets are proposed such as ImageNet [5], deep neural networks have been proven much more effective than traditional statistical learning method with hand-crafted features. For example, VGG networks [22] stacked multiple convolution layers to aggregate image spatial information. ResNet [9] and the following work ResNet-1k [10] have suppressed human performance on ImageNet image classification task.

Although CNNs have already achieved great performance, they were viewed as a black box and researchers thrived to understand and explain the internal mechanism of CNNs. The first step is to understand internal representation of convolutional filters by unpooling and deconvolution operations [26] [14] and arrange the network architecture for better higher representation [6]. Also, the abstract internal representation at higher level is obtained through DeepDream [15] to show the learned information and knowledge by CNN models. Apart from internal representation embodied inside CNN models, recent works include to visualize activation of a specific image, which is the CNN models' response to input images. The very first work is Classification Activation Map (CAM) [27] using the special property of sequential combination of global average pooling and output projection. Since this method has limits on usage, a more general method, Grad-CAM [21], is proposed to incorporate into any neural network architecture. Grad-CAM visualize the backpropogated gradients for each class rather than direct feedforward feature maps. After reviewing the previous visualization paper on CNNs, we found CAM was the most effective one and our architecture is compatible with this method.

3. Visualization Methods

In this section, we will introduce our visualization methods on internal representation for SVM with linear kernel, SVM with linear PCA. Note that the internal representation is learned on training dataset and independent with inference images. Then we will briefly talk about the activation map visualization method for linear SVM, linear SVM with linear PCA as well as CNNs.

3.1. Internal Representation Visualization

First, we will show our visualization for SVM with linear kernel. Recall that for a binary classification problem, the decision function for linear SVM is

$$\operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i^\top \mathbf{x} + \rho\right)$$

For multiclass classification setting, we use one-versus-rest (OVR), a.k.a. one-versus-all, mechanism [1]. In the OVR setting, we train C binary classifiers, where C is cardinality of category set $\mathbb{C}$. Hence, the decision function for OVR multiclass SVM with linear kernel is

$$\underset{c\in\mathbb{C}}{\operatorname{argmax}}\left(\sum_{i=1}^{n} \mathbb{1}_{y_i=c} \alpha_i \mathbf{x}_i^\top \mathbf{x} + \rho_c\right)$$

Here we define weight vector $\mathbf{w}_c = \sum_{i=1}^{n} \mathbb{1}_{y_i=c}\alpha_i \mathbf{x}_i$ for class c. Note that $\mathbf{w}_c$ should have the same dimension as vectorized image x. Since $\mathbf{w}_c$ affect the classification result for class c by dot product with input, we could consider the weight vector $\mathbf{w}_i$ as the internal representation of class c.

Apart from SVMs, we found a way to visualize the internal representation of linear PCA as well. Recall the PCA with learned parameter sets $\boldsymbol{\beta}$ works as follows. To project $\mathbf{x} \in \mathbb{R}^k$ into features $\mathbf{x}^p \in \mathbb{R}^k$,

$$x_i^p = \sum_{j=1}^{n} \beta_{ij} \mathbf{x}_j^\top \mathbf{x}$$

Similarly, we define $\mathbf{p}_i = \sum_{j=1}^{n} \beta_{ij}\mathbf{x}_j$. Consequently, linear SVM with linear PCA is

$$y = \underset{c\in\mathbb{C}}{\operatorname{argmax}}\left(\sum_{i=1}^{n} \mathbb{1}_{y_i=c} \alpha_i \mathbf{x}_i^\top \mathbf{p}_i^\top \mathbf{p}_i \mathbf{x} + \rho_c\right)$$

Here we define $\mathbf{w}_c^p = \sum_{i=1}^{n} \mathbb{1}_{y_i=c}\alpha_i \mathbf{x}_i^\top \mathbf{p}_i^\top \mathbf{p}_i$ and visualize $\mathbf{w}_c^p$ as the internal representation of linear SVM with linear PCA.

3.2. Activation Map Visualization

In addition to internal representation visualization, we find ways to visualize activation map from the trained models, which is the response for a specific from the model given one input image.

First we provide a generic activation map visualization method for both SVM and SVM with PCA. Generally, the decision function of both SVM and SVM with PCA is

$$y = \underset{c \in \mathbb{C}}{\operatorname{argmax}} \left(\mathbf{w}_c^{(p)} \cdot \mathbf{x} + \rho_c \right).$$

To obtain the feature map, we replace the dot product between weight vector $\mathbf{w}_c^{(p)}$ and vecotrized image $\mathbf{x}$ with Hadamard product, which is element-wise multiplication. Since all terms in the Hadamard product are summed up for final classification, the activation can be viewed as contribution to the final grade from each pixel. Hence, in the experiment, we plot the activation map as heatmap for explainable visualization. Note that we ignore the bias term ρ_c since it is applied to all classification and our activation map is normalized.

As for activation visualization of CNN models, Classification Activation Map (CAM) [27] is used for comparison study. CAM can be only used in the presence of global average pooling and output projection linear layer. The score for class c, p_c, which is an increasing function of the prediction probability is

$$\begin{aligned} p_c &= \sum_{i=1}^{m} \mathbf{w}_i^{c\top} \left(\sum_{x,y} \mathbf{F}_{x,y} \right) \\ &= \sum_{x,y} \left(\sum_{i=1}^{m} \mathbf{w}_i^{c\top} \right) \mathbf{F}_{x,y} \\ &= \left(\sum_{i=1}^{m} \mathbf{w}_i^{c\top} \right) \cdot \hat{\mathbf{F}}, \end{aligned}$$

where $\mathbf{w}_i \in \mathbb{R}^c$ is the i^{th} vector of weight matrix in the output projection layer, $\mathbf{F}$ is the feature map after final convolution and $\hat{\mathbf{F}}$ is the vectorized version of $\mathbf{F}$. Similarly, we replace the dot product with Hadamard product and obtain the activation map with same dimension as the provided feature map. Then we upscale the intermediate activation map back to the dimension of input image using bilinear interpolation.

4. Experiments

Procedure of our experiments[1] can mainly be partitioned into model training and model attribute visualization. For model training, our goal is to seek a model with highest test accuracy for each category of models.

[1] Our code is open sourced on GitHub

Model	Test Accuracy
Resnet-34	64.2%
Resnet-18	61%
VGG-13	61%
Adaboost	48.9%
Random Forest	48.6%
SVM + PCA	44.5%
SVM	41%

Table 1. Results of test accuracy

4.1. Dataset and Implementation

The task is based on a dataset of FER2013[2] consisting of 28,709 training images, 3,589 validation images and 3,589 test images. Each image is 48×48 grayscaled and labeled into 7 classes of different expression (Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral). The dataset was first introduced during ICML 2013 Challenges in Representation Learning.

Statistical methods, including SVM, PCA, Adaboost, Random Forest are implemented by `scikit-learn` api while Neural Networks are implemented in PyTorch[16]. Google cloud platform provides computation resources of CPU and GPU for our task.

Data augmentation is applied to neural networks. Since each image has a size of 48×48, the deeper neural networks is, the easier over-fitting happens, and there will be a great gap between training accuracy and test accuracy. Before each training epoch, every image is assigned with rotation of a random degree in range of $(-20°, +20°)$. Stochastic Gradient Descent with momentum [19] and Adam[12] are taken as optimization methods of neural networks.

4.2. Metric Results

Results of test accuracy are all shown in Table 1. The highest accuracy is achieved from ResNet-34. Among statistical learning methods, Adaboost gets better accuracy with Random Forest achieving a close result.

Here come some general results of parameters for statistical methods, as shown in Figure 1, 2, 3, 4, 5. Adaboost achieves best results with highest `n_estimators` and largest `max_depth`. However, increasing of these two parameters brings a trade-off between accuracy and computational cost. **Random Forest** shows better performance with `n_estimators` $\approx$ 150, `max_depth` $\approx$ 15 and `learning_rate` ≈ 0.1.

For SVM and PCA, we compared using Linear, Polynomial, Radial Basis Function (RBF) and Sigmoid kernels for SVM and SVM + PCA, and we hold `n_components` in PCA as 128. The compared result of accuracy and number of support vectors shown in Figure 6.The combined results show that SVM + PCA model with RBF kernel performed best with test accuracy of 44.2%. Additionally, we

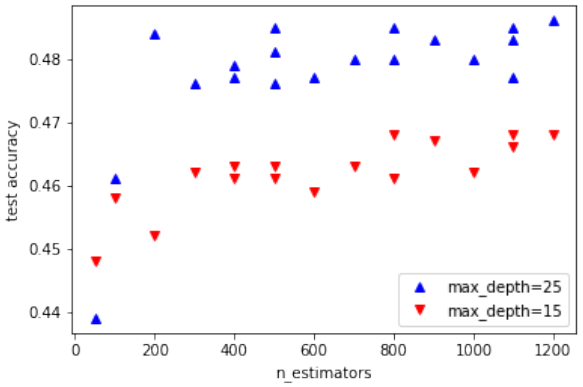

Figure 1. Adaboost: Test accuracy increases with n_estimators growing.

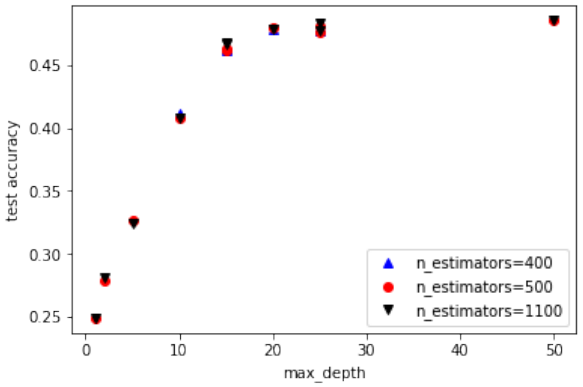

Figure 2. Adaboost: Test accuracy increases with max_depth growing.

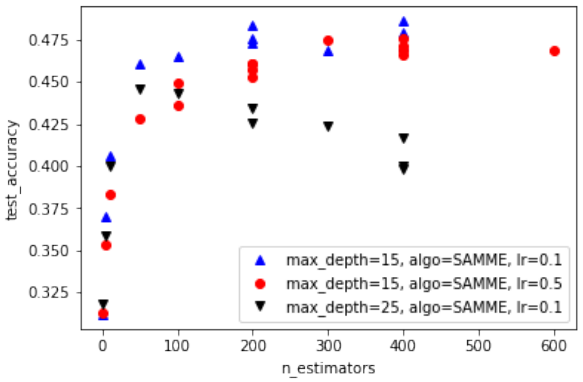

Figure 3. Random Forest: Test accuracy increases with n_estimators growing when n_estimators $\leq$ 200.

trained PCA + SVMs and Kernel PCA + SVMs models to see the effect of different choices of `n_components`. We preprocessed data for those two models with different `n_components` in $[50, 100, 200]$ as variables. The result of this experiment reveals that we got best performance when `n_components` is 50 used for PCA + SVM model.

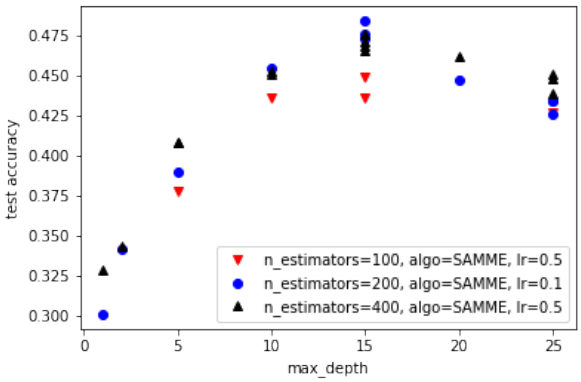

Figure 4. Random Forest: Test accuracy increases with max_depth growing when max_depth $\leq$ 15.

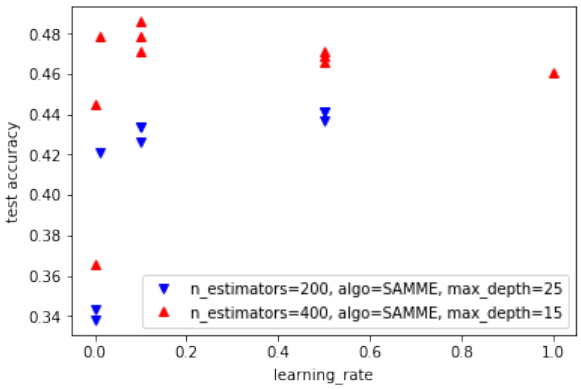

Figure 5. Random Forest: Test accuracy approaches the highest when learning_rate is near 0.1 .

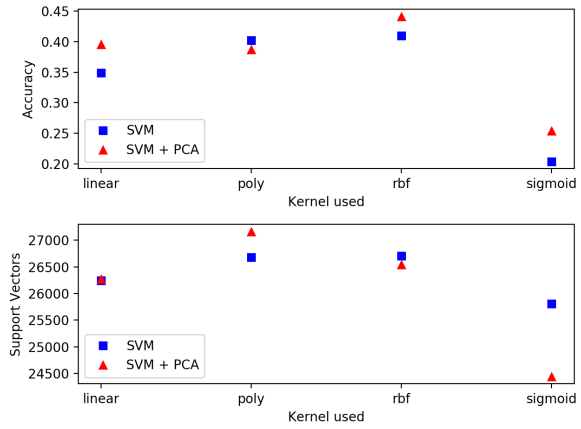

Figure 6. Support Vector Machine: Test accuracy and support vectors with different kernels used

The comparison for accuracy and number of support vectors are shown in Figure 7.

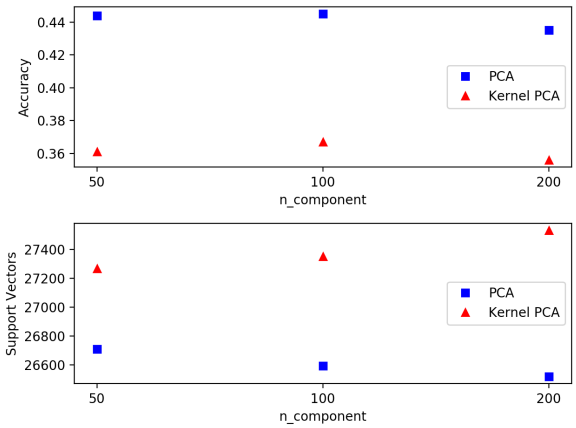

Figure 7. PCA: Test accuracy and support vectors with different n_components used

4.3. Internal Representation Visualization of SVM and PCA

First, we use SVM model and extracted features based on previous equations we derived. As you can observe from the internal representation for each class on the Figure 8, SVM considers each pixel independently but not local spatial features. However, we could still track subtle feature representation we got from SVM model in Happy and Surprise, for pixels near mouth. By using PCA + SVM, we have a significant extraction from our model. The feature maps observe very well. It is very clear to see each expression we extracted. More specifically, emotions have a great relevance with modes of mouth as suggested from the visualization results. Moreover, subtle texture near eyebrows and eyes indicates tiny differences between similar expressions. Comparing linear SVM with and without linear PCA, we can find out that the dimension reduction not only makes the computation cost lower but also smooth the feature map from weight vector.

Furthermore, we show several internal representations for kernelized PCA including RBF Kernel in Figure 9 and Sigmoid Kernel in Figure 10. Note that the internal representations, which is the inversed images, are provided as `inverse_transform` from the `KernelPCA` class [20] in the scikit-learn package. We plot feature representation with `linear` kernel in Figure 11 and found difference between two "Linear PCA" implementation. After reviewing the source codes, we argue that function `inverse_transform` is to calculate the following quardratic optimization problem [24],

$$z = \underset{z}{\operatorname{argmin}} \|\Psi - \phi(z)\|^2$$

which is only an approximation of inverse mapping from kernel space back to the original pixel space. Since sometimes accurate calculation of these inverse mappings is impossible (e.g. RBF kernel map vectors to infinite space), we only present our experiments here rather that compare them with CNNs in the following activation map comparison experiment.

4.4. Activation Map Visualization

According to the result of the activation map trained by ResNet-34 in Figure 12, class activation maps together with original images are listed in the order of its prediction score. We got the correct label with highest score. The red color on the map indicates the region has a very high activation and blue part represents low. The images with activation map show that the significant affect is made around mouth. Note that the heatmap is all normalized to [0, 1], which means the color only reflect relative magnitude of probability rather than prediction probability itself. Reversely thinking, red regions concentrate more on images of higher probability than on those on lower probability, which means mouth region indicates the expression best. By contrast, red regions scatter more on images of lower probability implies that useful features do not fit with the pixels.

Finally, we compared three models together, where all three models has succeeded to finding the feature representation of each expression. The results show that these models identified common elements or patterns based on seven expressions we gave. Given two images example shown in Figure 13, the most significant feature extracted to distinguish image is around the mouth which is the same procedure like humans.

5. Discussion

Both methods of feature extraction in SVM + PCA and CNN agree that mouth is the most important part for facial expression recognition, which is explicitly shown in proposed visualization result in both ways. And this conclusion is harmonious with our real-life experience. Moreover, visualization of SVM + PCA also tells that subtle texture difference near eyebrows and eyes are also important. Beyond our experiments, here is our thinking related to the overall implementation and visualization results of facial expression recognition.

First, our visualization for SVM + PCA can only be used in Linear SVM and Linear PCA. For Linear SVM part, we used an attribute `coef_` in SVC model which is only available in case of a linear kernel. For Linear PCA part, `PCA` in `scikit-learn` package is actually a linear PCA, having a great gap with `Kernel PCA` in implementation. The `inverse_transform` of Kernel PCA is based on an approximation method so we cannot obtain a weight vector for vectorized input images from Kernel PCA, making a constraint on our method.

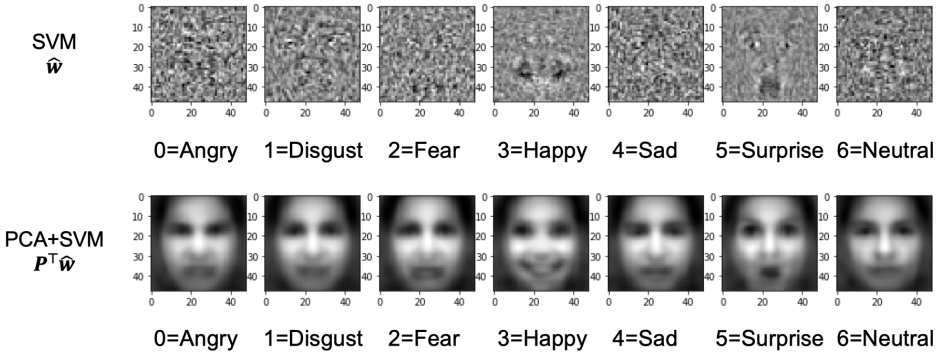

Figure 8. Internal representation of Linear SVM and Linear SVM + PCA. Note that PCA here references to a method of the exactly same name in scikit-learn library, not the Kernel PCA with linear kernel. Result of the latter one is shown in Figure 11, which shows worse performance in feature representation.

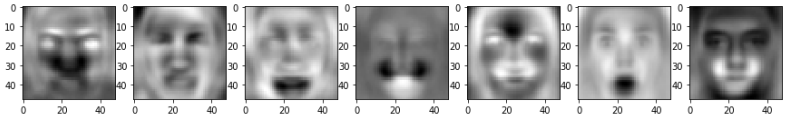

Figure 9. Feature extraction from Linear SVM + RBF Kernel PCA.

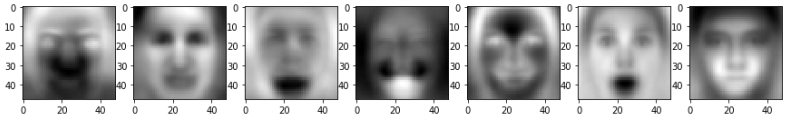

Figure 10. Feature extraction from Linear SVM + Sigmoid Kernel PCA.

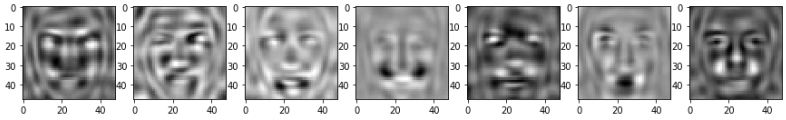

Figure 11. Feature extraction from Linear SVM + Linear Kernel PCA.

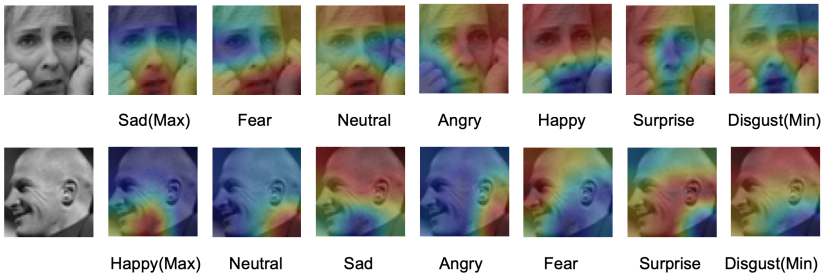

Figure 12. CNN: Activation map visualization

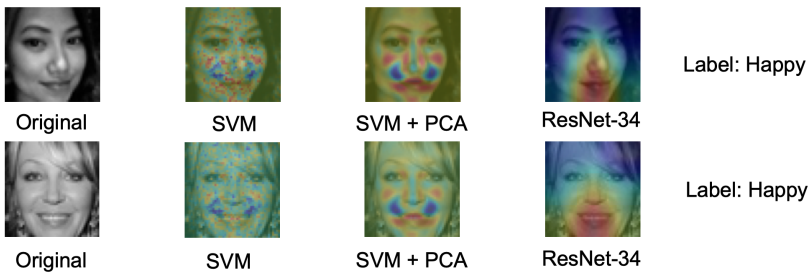

Figure 13. Visualization summary

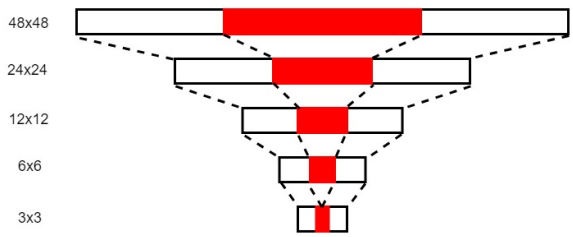

Figure 14. Correspondence relationship between the activation region on the first layer and one pixel on the last layer.

Second, visualization of activation map is limited by the input image size. After four max pooling layers in ResNet, each image of 48×48 is pooled into 3×3 as shown in Figure 14. As a result, when activating one pixel on the last layer, the corresponding activation region on the first layer is quite large relatively in the view of the whole image. However, we should better not pick a dataset of bigger image size because there exists a trade-off between computation cost and image size. If increasing image size, although we can have activation map visualization of CNN makes more sense, more time is needed to train SVM or SVM + PCA. Even with 48-by-48-pixel grayscale images currently, which are a quite small size in vision tasks, we still spend over hours to train a single model of SVM or PCA.

Third, recently we have noticed that there are some classic perspectives for facial expression recognition, including blocking certain parts as a metric of model robustness. It would be our future work since whether relationship of features has an impact on visualization method or not has not been tested.

In conclusion, our methods of visualization for SVM, SVM + PCA and CNN have shown interesting features. Deeper exploration into one field has much work to do, but raw mixture stills needs some related academic breakthroughs.

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] P.-L. Carrier, A. Courville, I. J. Goodfellow, M. Mirza, and Y. Bengio. Fer-2013 face database. *Universit de Montral*, 2013.

[3] J. Chen, Z. Chen, Z. Chi, and H. Fu. Facial expression recognition based on facial components detection and hog features. 2014.

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[6] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[7] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[8] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, et al. Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 64:59–63, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[11] T. K. Ho. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE, 1995.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[15] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. 2015.

[16] A. Paszke, S. Gross, S. Chintala, and G. Chanan. Pytorch, 2017.

[17] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[20] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Artificial Neural Networks — ICANN'97*, pages 583–588, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] N. K. Vasanth P.C*. Facial expression recognition using svm classifier. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, pages 16–20, 03 2015.

[24] J. Weston, B. Schölkopf, and G. H. Bakir. Learning to find pre-images. In *Advances in neural information processing systems*, pages 449–456, 2004.

[25] L. Xianwei and C. Guolong. Face recognition based on pca and svm. pages 1–4, 05 2012.

[26] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[27] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.