Feature Extraction by Deep Learning Methods

FANG Linjiajie (20382284), Liu Yiyuan (20568864), Wang Qiyue (20672641), Wang Ya (20549569)

Abstract

In this project, we explore the behaviour of feature extraction by deep neural networks on MNIST data set. Before we go through the process of validation of the neural collapse phenomenon, we firstly study features extracted by wavelets. We build a light scattering net coupled with 8 residual network blocks. Yet simple, after trained on one GPU for less than 17 minuets, the model can achieve over 99.5% accuracy on the test set. We study both the features extracted by wavelets and the features in the activation layer of our scattering net. We view the layer before the output, $\mathbf{x} \to \mathbf{h}(\mathbf{x})$, where $\mathbf{h} : \mathbb{R}^d \to \mathbb{R}^p$ outputs a p-dimensional feature vector, as the extracted features. We drop the softmax function in the final output and simply apply the linear classifier $\mathbf{W}\mathbf{h}(\mathbf{x}) + \mathbf{b}$ to do prediction, which is the same as the approach in the paper by Donoho (2020).

1 MNIST - hand written digits

We demonstrate our methods using the MNIST dataset. The dataset contains 70000 labelled 28×28 gray images with hand-written digits form 0 to 9. We follow the default train-test split given by Pytorch, which gives a training set with the size of 60000 and testing set with the size of 10000. Figure.1 gives 24 examples of those images:

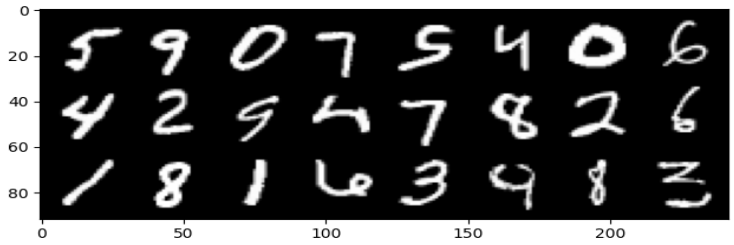

Figure 1: MNIST

In our project, we use a batch of 100 images to implement the training phase.

2 Feature extraction by wavelet scattering

2.1 Mother wavelets and scaling function

We use 2-dimensional Morlet filters to extract features. In our project, we only use 1st-order scattering for simplicity. Since our input image is of size 28×28, we limit the maximum scaling to be 2^2 so that the output features are of size $17 \times 7 \times 7$ with 17 channels and width&height equal 7 respectively. As for different direction, we use 8 angels ($\theta = 0, 1, ..., 7$) to perform the transform. Figure.2 shows the visualizations of the mother wavelets and the scaling function:

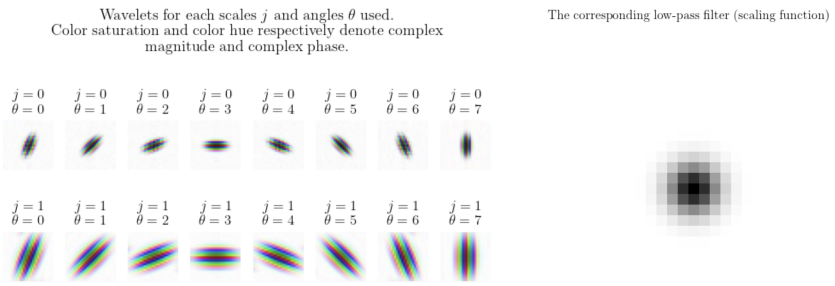

Figure 2: Mother wavelets and the scaling function used in our project

2.2 Scattering features of the image

Figure.3 shows examples of extracted features by the wavelets. Convolution w.r.t. the scaling function simply blurs the whole image. Different j stands for different scattering scales. A larger scale ($j = 1$) has a slightly stronger signal (brighter image) than the smaller one ($j = 10$), which might be the reason that the hand-written digits are comparably sparse: most of the nearby pixels of a signal are zero valued background. Also, obvious angle preferences are shown by comparing different θ. $\theta = 0$ emphasizes on the direction slightly to the right of a up-straight direction (see Figure.2), while $\theta = 1$ measures in a horizontal way. For instance, we can check the number '7' in the 6th column. In $\psi_{j,\theta=0}$, the vertical line is kept and the horizontal line is masked, while in $\psi_{j,\theta=3}$, the horizontal line is preserved and the vertical line is faded.

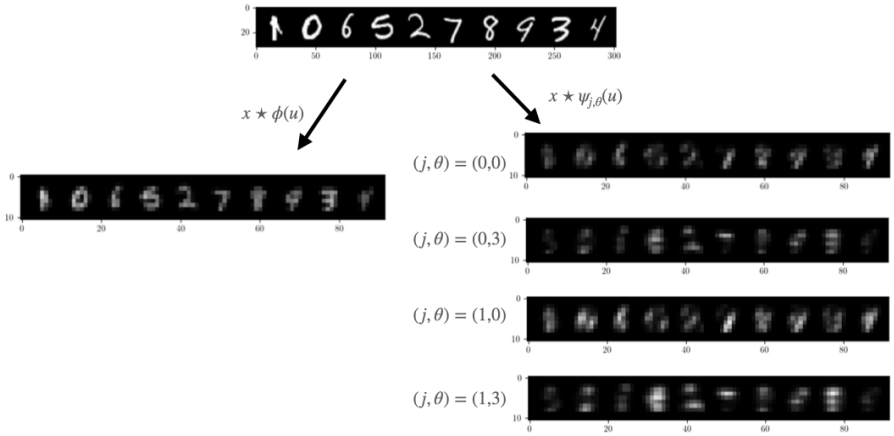

Figure 3: Examples of wavelets transformations of hand-written digits. Here we only show $\theta = 0, 3$ as examples

The total number of output channels is 17, which stands for features extracted from 8 angles in 2 scales separately (2*8=16) as well as one from the scaling function.

We can simply treat the wavelet scattered tensors as feature vectors of each image. In our case, the feature vectors have dimension equals to 17*7*7 = 833. To show whether such fixed wavelet transformation can directly give prediction ability, we visualize the 833-dim feature vectors using t-SNE. The result is given in Figure.4.

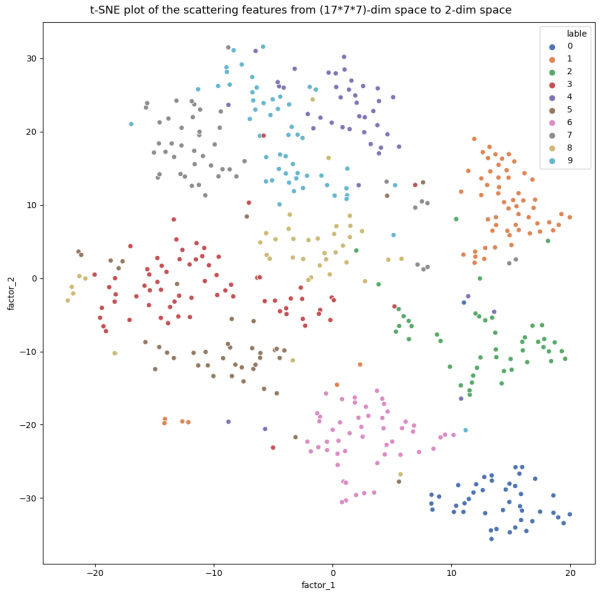

Figure 4: The t-SNE plot of the features given by scattering. It can be seen that, even though no train has been performed, the features already have power to classify different images. Notice that '0' can be successfully separated by a simple hyper-plane in 2-dim embedding of t-SNE. It can be explained by the feature of '0' that is more 'symmetric' in angles than other digits. The closeness between '0' and '6' is also reasonable by the fact that they are often similar to each other in hand-written images. There are only sample points plotted for illustration.

2.3 Feature extracted by other pre-trained deep methods

As comparison, we also use pre-trained ResNet18 and VGG11 to extract features of the hand-written data. ResNet18 and VGG11 outputs vectors of dim=1000. Figure.5 shows the result. Since the input images in MNIST only have 1 channel (gray image) while ResNet18 and VGG11 only accepts 3-channel RGB inputs, we make modifications of the original images by simply duplicating the channels by three times. After changing the hand-written digits to RGB-channels, we feed them to those deep networks.

3 Scattering net

3.1 Model architecture

After performing the scattering transform, each $1 \times 28 \times 28$ image (1 channel) will be transformed to a tensor of size $17 \times 7 \times 7$. We design a scattering network by feeding shape $17 \times 7 \times 7$ tensors to a deep residual network with 8 residual blocks and output a 10-dim vector for classification. In this section, we use $\mathbf{W}\mathbf{h}(\mathbf{x}) + \mathbf{b}$ as the linear classifier to make class prediction, i.e., we make prediction by $\text{argmax}_{c'} < \mathbf{w}_{c'}, \mathbf{h} > + b_{c'}$ and we simply treat $\mathbf{h}(\mathbf{x})$ as our activation vectors (or extracted feature vectors by the scattering network). This view of feature vectors is the same as Donoho (2020). In our experiment, the feature vectors are given by vectors of 512 dimensions.

The architecture of our scattering net is shown in Figure.6.

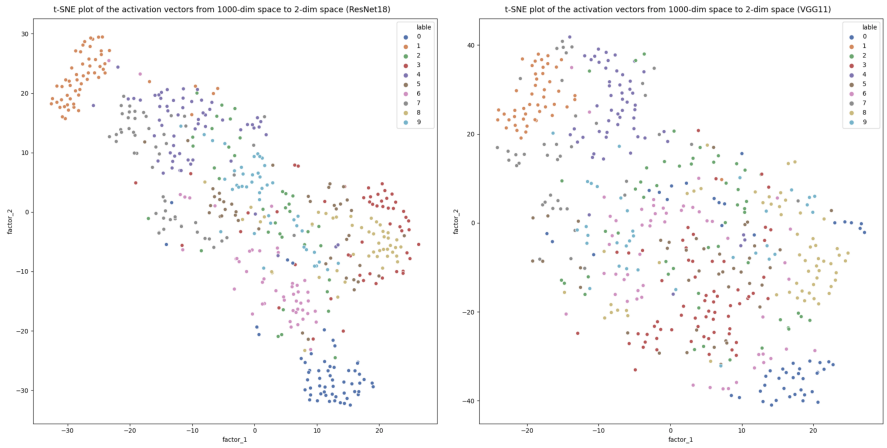

Figure 5: t-SNE plot of the features given by ResNet18 and VGG11. It can be seen that features extracted by ResNet18 can already have power to discriminate digits '0' and '1'. The features for VGG11 looks less distinguishable. The between-class gaps of two pre-trained models seem closer compared with wavelet transforms

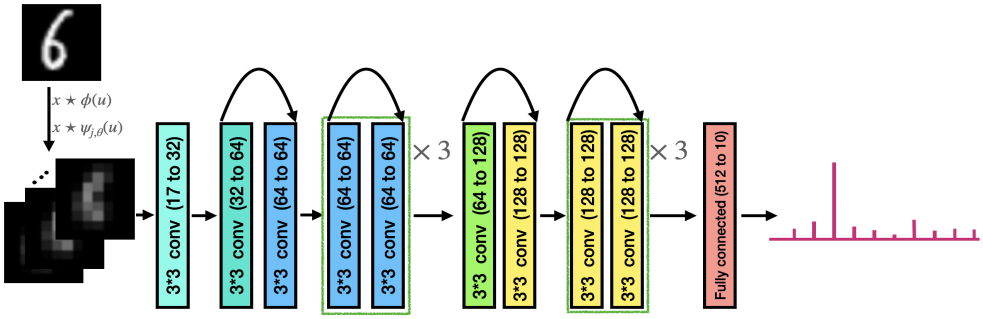

Figure 6: The architecture of our scattering net. After scattered using wavelets, we use 8 residual blocks with one fully connected layer to output the activation vector. Before the fully connected layer, the activation vectors are length with 512 dimensions

3.2 Performance

We train the model using cross entropy loss and stochastic gradient descent with the momentum=0.9 and the decay of weights = 0.0005. The learning rate is initiated to be 0.005 and annealed by a factor of 0.5 every 20 epochs. After trained by 100 epochs (with a single GPU with less than 17 minutes), the model achieves over 99.5% test accuracy. The overall performance is given in Figure.7

Since our initial learning rate is set to be high enough, both training and testing loss drop rapidly in the first few epochs. The second stage of rapid improvement appears when learning rate is annealed to be 0.0025 after 20 epochs. Further improvement to 99.5% accuracy is achieved after 40 epochs. Such control of the learning rate is effective to make the model training more stable.

3.3 Activation vectors of the scattering net

To visualize the activation vectors in the last layer (feature vectors) extracted by the scattering net, we use t-SNE to reduce the dimension from 512 to 2. Figure.8 shows the visualization of activation vectors after trained by the scattering net, the output activation vectors contract tightly to the class mean. The visualization pattern looks central symmetric, which suggests equi-angularity of the class vectors. Further validation of such neural collapse phenomenon is given in the next section.

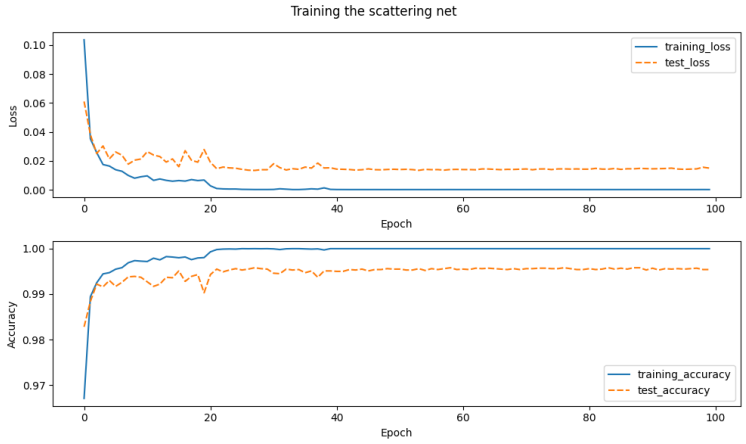

Figure 7: The model performance during the training and testing epochs. The loss drops rapidly at the very beginning. The later drop of the loss benefits from the annealed learning rate. After the 40 epoch, the model maintains 99.5% testing accuracy and zero training error.

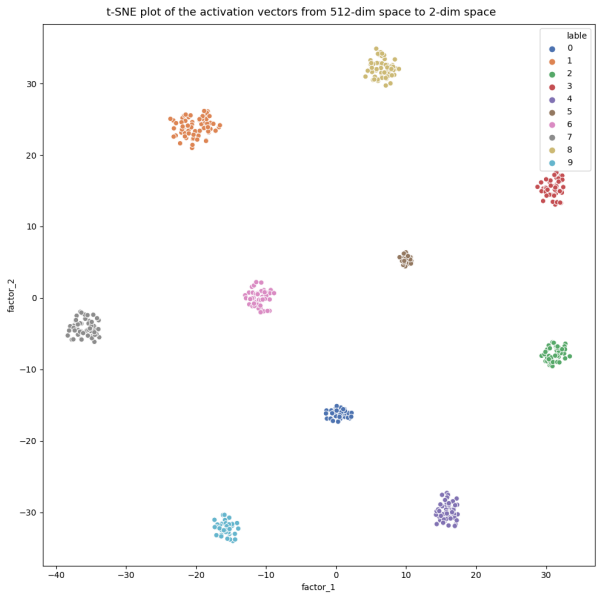

Figure 8: The visualization of the activation vectors extracted by our scattering net. Notice that the activation vectors are tightly contracted to each other and mutually symmetric around the center. There are 500 sample points plotted for simplicity.

4 Neural collapse phenomenon

Comparing Figure.4 with Figure.8, we have seen that the training points in the same class are contracted to the class mean, so it looks to agree with the NC1 that $\Sigma_w \to 0$. Additionally, in Figure.8, the central symmetry of the class clustering pattern gives insights into equi-angularity. To further demonstrate both NC1 and NC2, we apply our scattering net on MNIST dataset, record the activation vectors during the training epochs and calculate means, μ_G, μ_c, covariances $\Sigma_T, \Sigma_B, \Sigma_w$ and related quantities given in the paper by Donoho (2020).

Figure.9 shows the neural collapse phenomenon during the training phase.

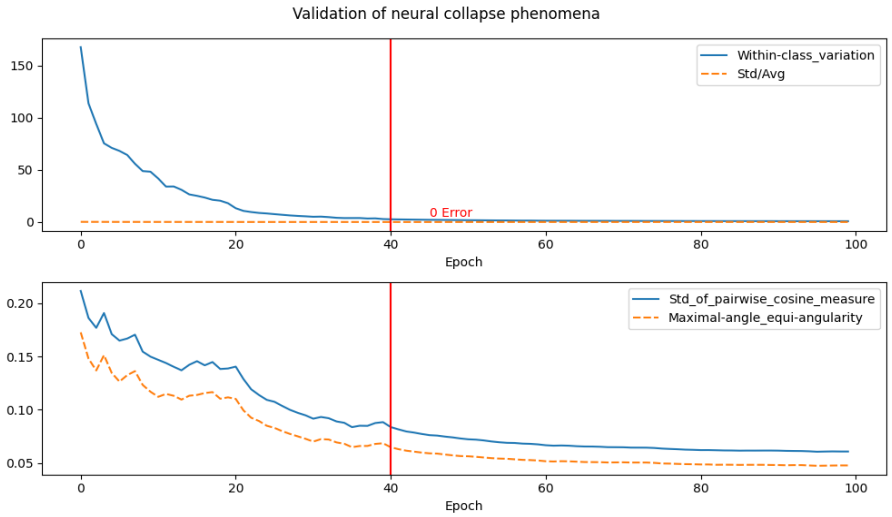

Figure 9: The neural collapse phenomenon. **NC1:** The upper figure shows that within-class covariance $\Sigma_w \to 0$ when the scattering net has zero training error. The solid blue line stands for $Tr(\Sigma_w \Sigma_w^T)/C$ and the dot orange line shows $\text{Std}_c||\mu_c - \mu_G||_2/\text{Avg}_c||\mu_c - \mu_G||_2$. **NC2:** The lower figure shows the equi-angularity of the extracted features. It's similar to the scenario of ResNet given by Fig.3 in Donoho's paper. The solid blue line is $\text{Std}_{c,c\neq c'}(cos_\mu(c,c'))$ and the dot orange line is $\text{Avg}_{c,c'}(|cos_\mu(c,c'), 1/(C-1)|)$. As training progresses, the Std of cosine measures closes to zero, indicating equi-angularity.

5 Extracted features used in traditional supervised learning methods

From the former sections we have used wavelets, pre-trained deep networks and trained scattering net to extract features. The pure wavelet transformation is simple and speedy for extracting features. In this section, we apply wavelet transformed features to train traditional supervised learning methods: logistic regression, Adaboost, Random Forest and SVM and focus on the activation vectors given by those methods. Figure.10 gives the visualization of activation vectors from different methods.

6 Conclusion

In this project, we explore the feature extraction by both scattering and learning methods. The visualization shows that wavelet transformed features already exist power of prediction. we train scattering net based on these features and partially reproduce the neural collapse phenomenon (NC1, NC2) in the paper by Donoho (2020). Additionally, we compare traditional supervised learning methods with deep methods and find that extracted features from traditional methods do not hold the property similar to neural collapse.

7 Contributions

- FANG Linjiajie: Programming and network designing
- Liu Yiyuan: Data preparation and figures drawing
- Wang Qiyue: Visualization of results and traditional methods
- Wang Ya: Outcome analysis and summarising
  All authors contributes to the report-writing.

References

[1] Vardan, P. , X.Y.Han, David L. Donoho (2020) Prevalence of Neural Collapse during the terminal phase of deep learning training

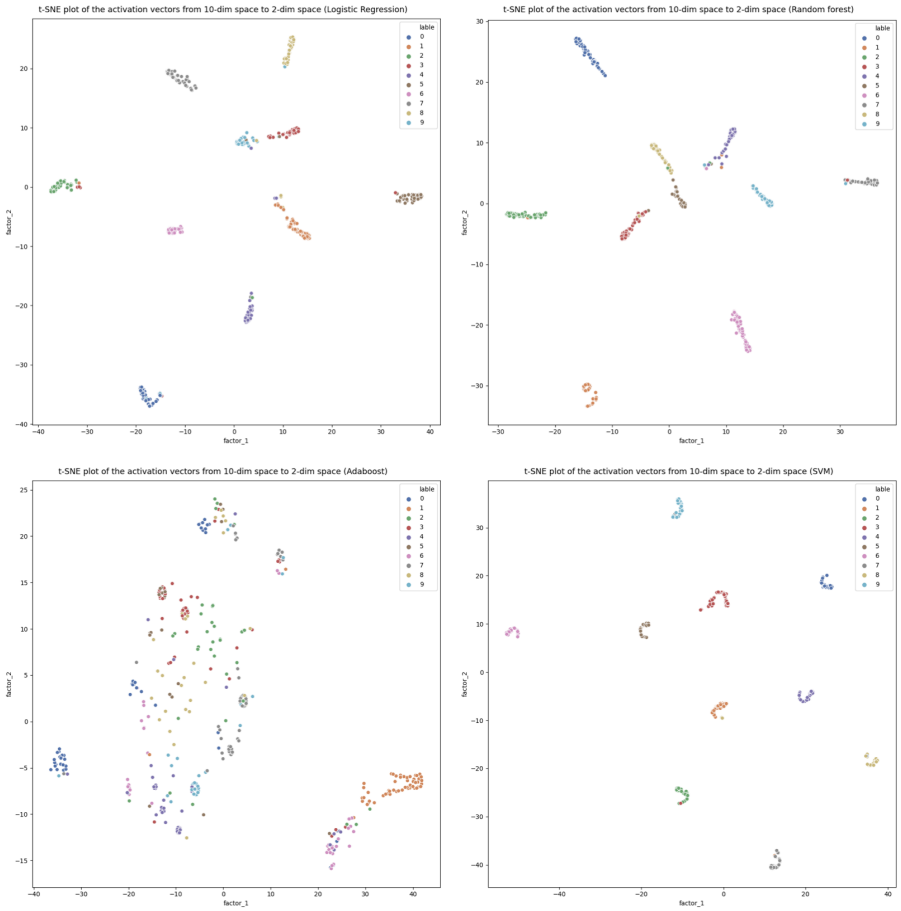

Figure 10: The visualization of the activation vectors by traditional supervised learning methods. They are trained by scattering transformed data. The accuracy in the training phase is: logistic regression=97.7%, random forest=100%, Adaboost=72.9%, SVM=99.0%. We mainly focus on the extracted features and train the model to the least training error. It can be seen that the activation vectors are less symmetric compared with scattering net. Feature vectors in logistic regression and random forest seem to be compressed to hyper-planes in their own classes. SVM gives class features in small curved manifolds. Adaboost are not trained well for the task and has difficulties in discriminating some close digits, such as '8', '6' and '2'. We see that there is no obvious equi-angularity phenomenon in these features. Additionally, unlike contracting feature vectors to vertices of a simplex, those features are most likely contracted to several class-specified manifolds.

[2] https://github.com/kymatio/kymatio (Scattering implementation in python)