Towards Better Understanding of Deep Learning and Scattering net with Feature Visualization

Tony C. W. Mok
Department of Computer Science
cwmokab@connect.ust.hk

Jierong Wang
Department of Computer Science
jwangdh@connect.ust.hk

Abstract

In this project, we aim to explore the feature visualization and neural collapse phenomena in deep learning-based method. Specifically, we utilize the scattering net and pre-trained VGG-19 to perform feature extraction on the two datasets MNIST and Fashion-MNIST. Feature visualization is performed on the features obtained from VGG-19 and Scattering Net respectively using PCA, t-SNE, LDA and ReliefF. After that, we study the neural collapse phenomena of deep learning by observing the magnitude of the between-class covariance compared to the within-class covariance of the train activation, the coefficient of variation of the centered class-mean norms, the standard deviation of the cosines between pairs of centered class-means and the average shifted cosine similarity between pairs of centered class-mean. Finally, we report the image classification result on the MNIST dataset using a bunch of conventional supervised classifiers.

1 Dataset

The MNIST dataset [2] is a database of handwritten digits, which consists of a training set of 60,000 images and a test set of 10,000 images. Each image is a single channel with 28×28 pixels. The MNIST dataset is a popular database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting. Figure 1 shows some example images in the MNIST dataset. In total, there are ten classes, i.e. 0-9, in both the training set and the testing set.

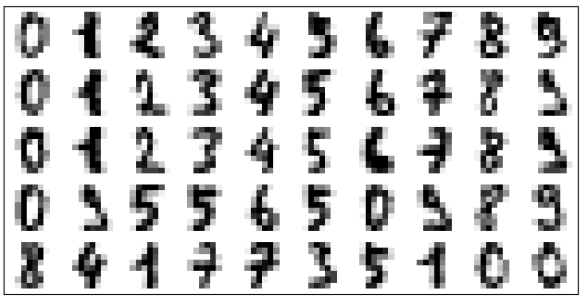

Figure 1: Example images from the MNIST dataset.

2 Feature Extraction

In this section, we focus on the feature extraction using scattering net [1] and VGG-19 [12]. We describe the details of each method below.

2.1 Scattering Network

The scattering network [1] constructs the invariant features by using predefined wavelet functions convolve with the input image in a multi-scale scenario. In the following experiments, the number of scale in the filter bank is 4 and the number of scale is 1 per octave. The wavelet function is rotated by 8 times with angle-step equals to $\frac{\pi}{8}$. The scat function will compute scattering coefficient of order 0 to 2 and an oversampling factor of 1^2. With the above construction, a 417×1 feature vector is generated for each image. Figure 2 shows the filter bank used in our experiments. We employ the ScatNet of Matlab version (Ref: https://www.di.ens.fr/data/software/scatnet/) in our project.

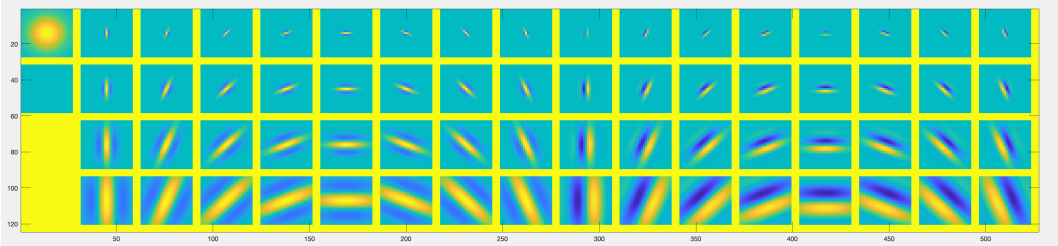

Figure 2: The filter bank used in our project for scattering network.

2.2 Pre-trained VGG-19

The VGG-19 [12] is a convolutional neural network with 19 learnable layers, including the fully-connected layers and convolutional layer. As training a VGG-19 from scratch could be time-consuming and may be suffering the over-fitting issue for a small dataset, we decide to adopt the pre-trained VGG-19 model to extract features. Specifically, we implement the pre-trained VGG-19 using Pytorch [10] and downloaded the pre-trained model from `https://pytorch.org/hub/pytorch_vision_vgg/`. This pre-trained VGG-19 model was trained with the database from the 2014 Imagenet ILSVRC challenge, which consists of 1000 classes of object and an average of over five hundred images per class.

By default, the pre-trained VGG-19 accepts 3-channel (RGB) input with resolution 224×224 and expects that there are 1000 classes for the dataset. Since we are using the MINST dataset (i.e. 10 classes and 28×28 pixels), we have to modify the pre-trained VGG-19 network or the dataset we used. To maintain the integrity of the network, we upsample the input images to 224×224 using bilinear interpolation. Then, we duplicate the input two times and concatenate them to form a 3-channel input before feeding the input images into the pre-trained VGG-19. Eventually, we can obtain a feature vector with size 4096 for each image in the MNIST dataset.

3 Feature Visualization Methods

To visualize the high dimensional vectors, feature selection techniques are required, which also can be regarded as dimensionality reduction. Given a large dimensional feature space, some dimensions may be dependent on others and some may be irreverent. With the help of feature selection, redundant dimensions can be discarded and thus reduce the data size and improve the efficiency and the accuracy of the training model.

In this project, we employ four feature selection methods to visualize the extracted features, including principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE), linear discriminant analysis (LDA) and ReliefF.

3.1 Principal Component Analysis (PCA)

PCA [14] unitizes the covariance matrix to perform dimensionality reduction. The first two dimensions are selected for visualization in the 2D space. However, since the covariance matrix only measures the linear relevance between two dimensions, the visualization effects may not be perfect for the nonlinear cases.

3.2 t-distributed Stochastic Neighbor Embedding (t-SNE)

The t-SNE [8] is a machine learning technique for non-linear dimensionality reduction, which is particularly suitable for the visualization of high-dimensional datasets. Throughout minimizing the Kullback–Leibler divergence between the probability distribution over pairs of high-dimensional objects and the probability distribution over the points in the low-dimensional map, we can visualize and separate the data points in low-dimensional space. For t-SNE, we implement two versions of it, one using the "manifold" package from the Sklearn library, the other one is the built-in t-SNE in Matlab.

3.3 Linear Discriminant Analysis (LDA)

LDA [15] is a kinds of supervised models. The basic idea of LDA is to project the high dimensional data samples onto the best discriminant vector space, in which the samples have largest inter-class distance and lowest intra-class distance. The main difference between PCA and LDA is that PCA is an unsupervised feature selection method and the number of reduced dimensions is relative to the number of features. On the other hand, the number of dimensions reduced by LDA is dependent on the number of labels.

3.4 ReliefF

ReliefF [11] algorithm is a feature weighting algorithm, which will assign a low weight on the dimensions that increase intra-class difference and assign a large weight on the dimensions that enlarge inter-class difference. We observe that apply the above two supervised methods directly cannot generate good point clusters under different classes, which exhibit over-linear phenomena. Therefore, t-SNE is employed before using the supervised feature selection methods, i.e.: LDA and ReliefF.

4 Feature Visualization Results and Discussion

In order to reduce the execution time and produce a clear feature visualization, we sample 10,000 images from the training set of the MNIST dataset. Then, we use the scattering net and the pre-trained VGG-19 to extract the feature. For VGG-19, we use "register_forward_hook" function in Pytorch to hook the output activation of the second fully-connected layer in the classifier module as the feature vector. Note that in the default settings, the classifier of the VGG-19 has 3 fully-connected layers.

Figure 3 and Figure 4 show the results of the scattering network features and VGG-19 features with different dimensionality reduction methods using the Matlab built-in library and the Matlab community (LDA, Ref: https://www.mathworks.com/matlabcentral/fileexchange/29673-lda-linear-discriminant-analysis). Each image feature corresponding to a color point in the 2D-space. The color representing the class of the data point (i.e. 0-9).

4.1 Scattering Net

We observe that t-SNE with LDA is the best for minimizing the within-class distance and maximizing the inter-class distance, while PCA is the worst method in terms of minimizing the within-class distance and maximizing the inter-class distance for features extracted by the scattering net. The pure t-SNE and t-SNE with ReliefF has similar performance.

4.2 Pre-trained VGG-19

The LDA fails to perform dimensional reduction on VGG-19 features and thus only three visualization results are shown for the pre-trained VGG-19. It is because for some dimensions in pre-trained VGG-19, zero within-class variance exists.

Yet, similar to the case in scattering net, t-SNE and t-SNE with ReliefF are able to minimize the within-class distance and maximize the inter-class distance, as shown in the Figure 4. Although PCA is simple and efficient, PCA remains the worst method for dimensionality reduction. It further implies that PCA may not be a good dimensionality reduction method for high dimensional data.

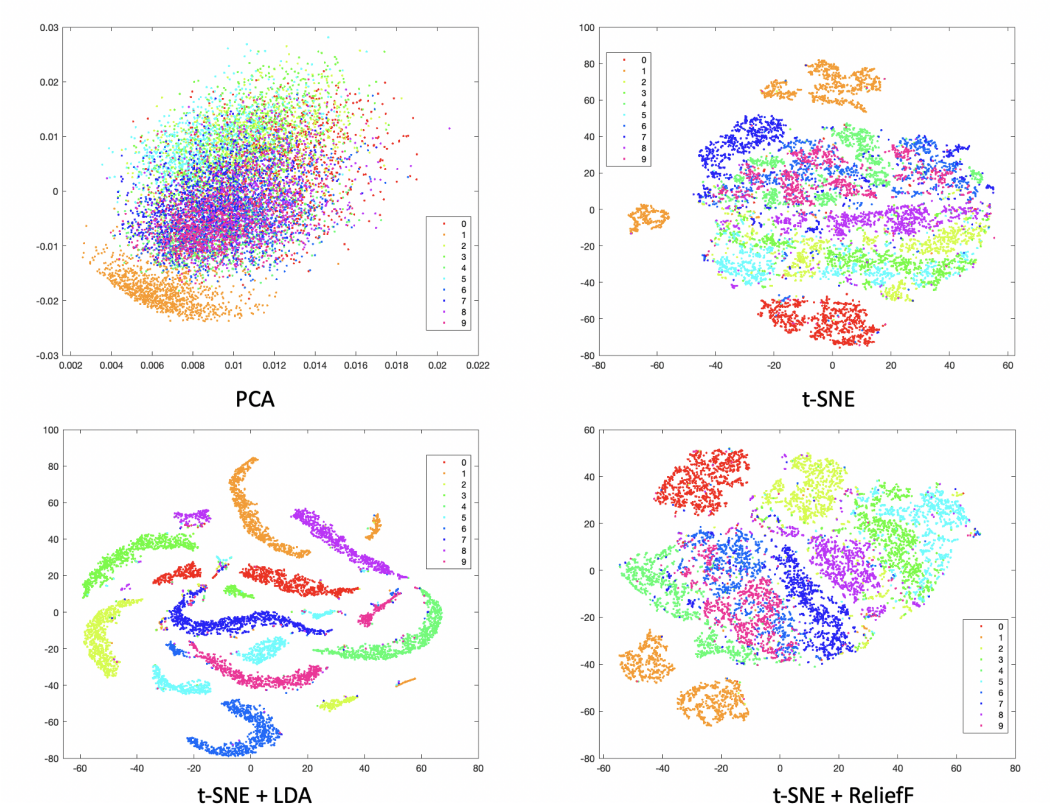

Figure 3: The feature visualization of scattering net.

Figure 5 shows the t-SNE results for both VGG-19 and scattering net using the Sklearn library with Python. It is interesting that the hand digits with similar visual appearance tend to be closer in the visualization of the features from VGG-19. For instance, digit "0" (colored in red) and digit "6" (colored in brown) are close in distance.

Furthermore, comparing the t-SNE results of VGG-19 and scattering net, we found that the features generated by VGG-19 tend to separate better than the one in the scattering net. For instance, the features of the digit "6" are mixed with the features of digit "9" in t-SNE (scattering net).

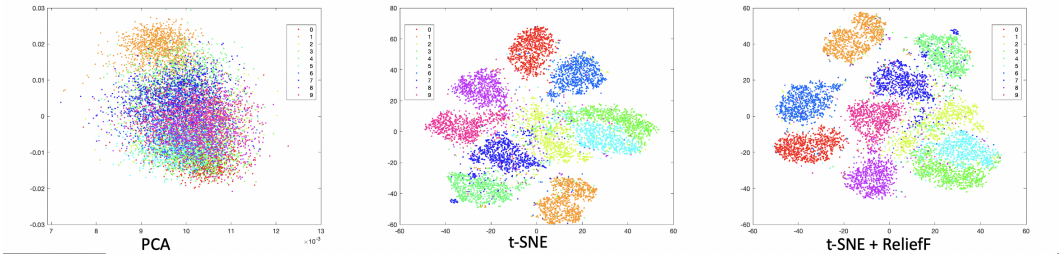

Figure 4: The feature visualization of pre-trained VGG-19.

5 Verification of Neural Collapse

To verify the neural collapse phenomena mentioned in [9], we follow the experiments in [9] to measure the magnitude of the between-class covariance compared to the within-class covariance of the train

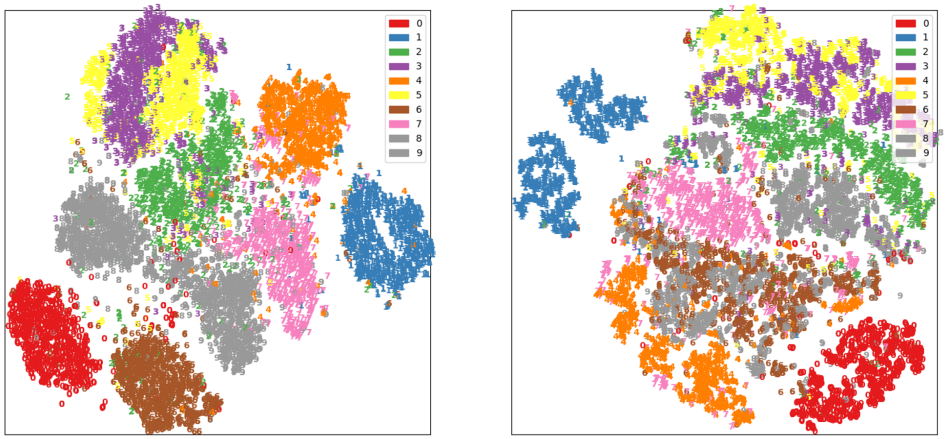

Figure 5: The t-SNE feature visualization of pre-trained VGG-19 and Scattering Net.

activation, the coefficient of variation of the centered class-mean norms, the standard deviation of the cosines between pairs of centered class-means and the average shifted cosine similarity between pairs of centered class-means. We compute and record these metrics for every 5 epochs during the training. In total, we fine-tune the pre-traine the last three fully connected layers in VGG-19 model for 350 epochs with Adam optimizer [7] (learning rate = $1e^{-4}$). Similar to the last section, we use "register_forward_hook" function in Pytorch to hook the output activation of the second fully-connected layer in the classifier module as the feature of the input data.

For the setup of the experiment, we highlight that there are differences between our setup and the setup in [9]. First, we employ a pre-trained VGG-19 and we only finetuned the classifier, i.e. the last three fully connected layers. Second, we use a smaller sample size (2000 images in the training set) for this task. Third, instead of upsampling the input image in order to fit the pre-trained network, we remove the last three max-pooling and use the raw training image for training. This change can greatly speed up the training process.

Although all the metrics show similar trends as in [9], unlike the experiment results in [9], all the metrics in Figure 6 do not coverage near zero. We conclude that there are three reasons. First, the sample size of our experiment is not as large as in [9] such that the outliners in the training set could bring a relatively huge impact to the total covariance matrix, between class covariance and within class covariance. Second, the sample size for each class is not identical in our experiment. The network may bais to some of the class during training. Finally, we are using a pre-trained network and finetune only the last three fully connected layers. Hence, the features from our model may not be the "perfect" feature for the MNIST dataset.

5.1 Within class variation

$$\frac{\mathrm{Tr}\{\sum_W \sum_B^{\dagger}\}}{C} \tag{1}$$

The first line graph in Figure 6 illustrated the magnitude of the between-class covariance compared to the within-class covariance of the train activations. Specifically, this is represented by the equation in 1, where $\sum_W$ and $\sum_B^{\dagger}$ denotes the within-class covariance of the features of the training data and the Moore-Penrose pseudoinverse of the corresponding between-class covariance. As shown in the figure, this value rapidly decreases from around 2.0 to 0.3 during the first 50 epochs. This indicating that during the training, the within-class covariance of the features is decreasing while the corresponding between-class covariance is increasing. We can draw the same conclusion as in [9]

that the within-class variation collapse during training and the within-class variation of the activations becomes negligible as these activations collapse to their class-means.

5.2 Equal-norms of class-means

$$\frac{\text{Std}_c(\|\mu_c - \mu_G\|_2)}{\text{Avg}_c(\|\mu_c - \mu_G\|_2)} \tag{2}$$

The second line graph in figure 6 depicts the coefficient of variation of the centered class-mean norms as shown in equation 2, where μ_c and μ_G represent the class-means and global mean of the features respectively. During the training, the value of the equation drop from 0.32 to 0.09 by the end of the training. This implies that the train class-means become equal-norms.

5.3 Equal-angularity

$$\text{Std}_c(\cos_\mu(c, c')) = \text{Std}_c(\frac{< \mu_c - \mu_G, \mu_{c'} - \mu_G >}{\|\mu_c - \mu_G\|_2 \|\mu_{c'} - \mu_G\|_2}) \tag{3}$$

The third line graph in figure 6 depicts the standard deviation of the cosines between pairs of centered class-means as shown in equation 3. During the training, the value of the equation gradually decreases from 0.35 to 0.15 by the end of the training. The standard deviations of the cosines approach zero indicating equal-angularity.

5.4 Maximal-angle equiangularity

$$\text{Avg}_{c,c'} |\cos_\mu(c, c') + \frac{1}{C-1}| \tag{4}$$

The rightmost line graph in figure 6 depicts the average shifted cosine similarity between pairs of centered class-means as shown in equation 4. During the training, the value of the equation gradually decreases from 0.344 to 0.131 by the end of the training. As this value goes toward zero during the training, it implies that all cosine similarity converges to $\frac{-1}{C-1}$. This corresponds to the maximum separation possible for globally centered, equiangular vectors as observed in [9].

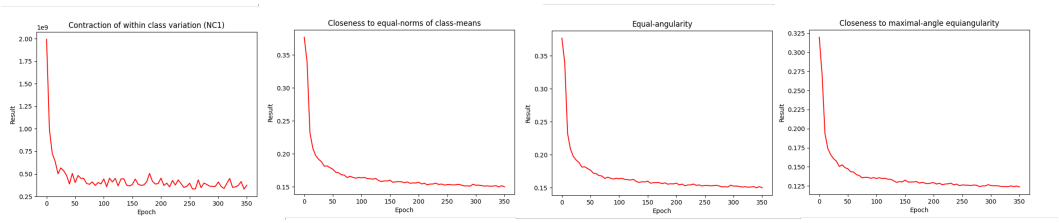

Figure 6: From left to right: the magnitude of the between-class covariance compared to the within-class covariance of the train activation, the coefficient of variation of the centered class-mean norms, the standard deviation of the cosines between pairs of centered class-means and the average shifted cosine similarity between pairs of centered class-means.

6 Image Classification on MNIST

We employ three-layer BPNN [5], SVM [13], KNN [6], decision tree [3] and Adaboost [4] for the classification task. In the BPNN, the number of hidden layers is 15 and the maximal iteration number is 550. The lambda is 0.1. For the SVM, we use C-SVM, which is a multi-class classifier and we specify the cost of C-SVM as 10. The kernel is the polynomial function $(\gamma * u' * v + \text{coef0})^d$, where $\gamma = \frac{1}{k}$, k is the number of features, coef0 $= 1$ and the degree $d = 3$. Regarding to the KNN model, We specify the number of nearest neighbors in the sample matrix as 1 for classifying each point when predicting. For the rest two models, we keep the default parameter setting during training. Table 1 shows the summary of the learning models and the hyper parameters we used for this task.

Table 1: The summary of learning models

	BPNN	SVM	KNN	Decision Tree	Adaboost
Source	Implemented by ourselves	LIBSVM	Matlab built-in function	Matlab built-in function	Matlab built-in function
Function Name	BPNN	SVM	fitcknn	fitctree	function

Parameter	hidden_layer_size = 15;
	num_label = 10;
	lambda = 0.1;
	max_iter = 550;
		'-s 0
		-c 10
		-t 1
		-g 1
		-r 1
		-d 3'
			'NumNeighbors',1	default	default

Table 2 and Figure 7 exhibit the classification results regarding to the accuracy of the test set. As shown in the table and figure, BPNN achieves the best score on both scattering network and VGG-19. followed by the SVM model. The performances of the tree-structure models such as decision tree and adaboost are not satisfying comparing to the other learning methods. It is interesting that both the learning features (VGG-19) and the non-learning features (Scatnet) have similar performances on each classifier. MNIST is a simple dataset and the classification task on such dataset is not a complicated task. Using the deep neural network may over-fit to the train data. Besides, although the size of each image in MNIST is 28*28, up-sampling is required before feeding into the neural network, which may have negative influence on the classification quality.

Table 2: The classification accuracy

	BPNN	SVM	KNN	Decision Tree	Adaboost
Scatnet	0.9843	0.9645	0.9408	0.8556	0.8666
VGG-19	0.9819	0.9817	0.9366	0.8211	0.8792

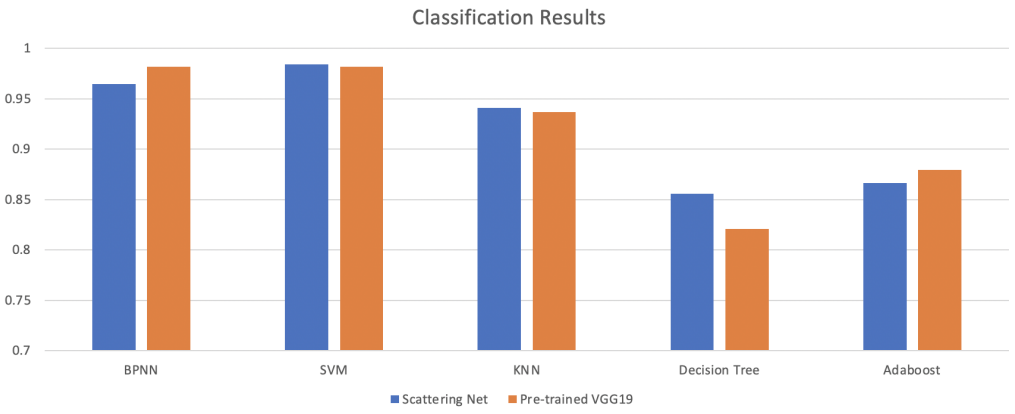

Figure 7: The classification accuracy.

7 Conclusion

This research project studied the feature visualization and neural collapse phenomena in deep learning-based method. Among the four dimensionality reduction methods, we found that t-SNE is the most robust method in visualizing the features from both VGG-19 and scattering net. Moreover, we studied the neural collapse phenomena through a sequence of metrics. We concluded that the ideal variability collapse exists if and only if (1) the training data for each class are even; (2) the sample size of the

training data is large enough; and (3) the features has high similarity for within-class data while has low similarity for inter-class data. Finally, we also examine the classification accuracy with different classical classifiers. The classification accuracy of BPNN and SVM outperformed the other classifiers in general, although the difference is small.

8 Individual Contribution

- **Tony C. W. Mok**: Report writing; VGG-19 feature extraction and visualization; Image classification on MNIST; Verification of Neural Collapse
- **Jierong Wang**: Report writing; Scattering Net feature extraction and visualization; Image classification on MNIST; Verification of Neural Collapse

References

[1] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[2] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[3] Mark A Friedl and Carla E Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.

[4] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[5] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[6] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4):580–585, 1985.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[9] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 2020.

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[11] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[14] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[15] Jieping Ye, Ravi Janardan, and Qi Li. Two-dimensional linear discriminant analysis. In *Advances in neural information processing systems*, pages 1569–1576, 2005.