NNS : Neural network surgery

S.Emadi[a,1], Zahra Davardoust.[b]

[a] *Faculty of Electrical and Computer Engineering University of Tabriz, Tabriz, Iran, s.emadi@tbz.com*
[b] *Faculty of Electrical and Computer Engineering University of Tabriz, Tabriz, Iran, z.davardoust@tbz.com*

Abstract

Transfer learning has emerged as a pivotal approach in machine learning, enabling models to leverage knowledge from one domain and apply it to another, often related, domain [22]. Methods like : **Pre-trained Models, Feature Extraction[7],Fine-tuning [4],Multi-task Learning,etc.** The use of these methods due to **domain dependence, complete and complex transfer of learning causes lack of understanding of the model, high computational cost, negative transfer and very limited application** of our model. With *neural network surgery (NNS)* and its in-depth investigation, we provide general knowledge about the specific learned topic with full coverage of the topic without bias and inter-neural interference in the process of learning transfer. *In this method, by taking a deep look at the events in the memory during calculations, as well as labeling them, stimulating the neural network case by case, and comparing the results, we tried to discover and limit the network to that particular label*. By removing the less important connected neurons to the rest of the labels, we have been able to prevent overfitting and also prepare general learning for transfer. By performing s*urgery on the neural network* that has learned the *MNIST [3; 8]*, we have extracted learning in a special way to recognize "3" labels. **Finally, it is possible to transfer learning to the next generations in a simple and easy way and to use it for continuous learning of special labels, and it is also possible to mention the transfer of learning special labels from very complex models with many labels to small models with High accuracy** In this research, new interesting results about the compression of neural network models are mentioned.

Keywords:
Transfer Learning, Fine-Tuning, Multi-Task Learning [21], neural network, Feature extraction, MNIST, Computer vision, CNN, NN architecture.

1. Introduction

The goal of learning neural networks is always to correctly classify the machine using learning through learning examples. Like a child, the more our learning examples are massive and the more parameters we have, the more accurate our learning will be [6]. Obviously, neural networks that include many parameters cause the formation of a heavy network that requires a lot of resources to go through the learning stages and perform many calculations. On the other hand, there are large and learned models that have obtained results by consuming resources. Generally speaking, they all need/have access to a lot of resources and data that most researchers don't have access to. For this reason, we have tried to track the prediction path and the paths of connecting layers and neurons to each other when re-categorizing the learned patches with the data of the same label (the data used for learning provide the best results), extract a subnet (Fig. 1) from the existing main neural network By doing this, by using the learned model (instead of learning the new model again), we can provide a new model with the least resource consumption and the simplest and lightest calculations. Also, because the general knowledge about the category of that tag is learned, we can personalize it for specific and relevant

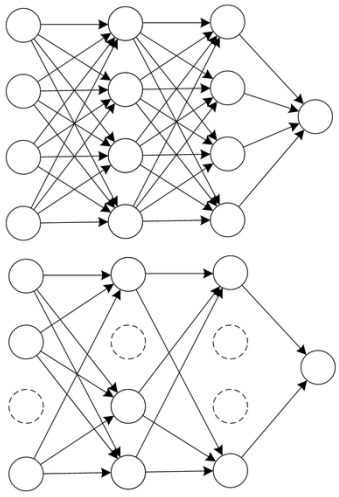

Figure 1: This figure shows that by removing the irrelevant connections, the nature of the model and classification remains the same (the model is lighter and pruned only to classify a particular label).

Email addresses: saeidemadi311@gmail.com (S.Emadi), za.davardoust@gmail.com (Zahra Davardoust.)

uses (after this proccess, can fine-tuning and transfer learning). It should be noted that in this method the previously learned model architecture will be preserved. In this article, we have tried to extract the labels classification sub-network from the learned MNIST labels model in this way.
In fact, we can make the most optimal use of this method to compress neural networks. This extraction of subnetworks helps us to reintegrate the subnetworks to arrive at a larger network, which is smaller than the original trained model, but still has significant accuracy, which makes us consider edge computing in the future.

2. Background

In the basic methods of transfer learning and fine-tuning, the original neural network is always discussed and calculated, which consumes too many resources. Also, we will not have the ability to use complex and large trained models for use in specific cases, or we will need a lot of resources and heavy calculations, which have been solved with this method [23; 20].
There are also methods that are close to this method The separable and interpretable network is for deep neural networks, which is able to show the functional behaviors of neural networks, which mostly focus on detecting and identifying fake samples [19].
Strategies for training neural networks with dynamic routing are also used in multitask learning, where graphs of learned formations are drawn through which different input signals take different paths where they find dynamic routing networks which are trained to classify images, layers and branches. But it is not possible to extract and use the related subnet with this method. Rather, it is used only to identify the forecast path [11; 13].
It should be noted that in the previously mentioned and tried methods, they focus more on connecting small networks to each other or predicting in parallel, while we have focused on the simplest and most basic method for extracting sub-netwrok of the main networks [16].
One of the first studies to use magnitude-based pruning for deep networks is a method that process in which weights below a given threshold are pruned. Once pruned, the network is fine-tuned and the process repeated until its accuracy begins to deteriorate[10]. Our work is more similar to this method, but in the form of semantic pruning.
Another method Data dependent channel pruning methods, which are based on the view that when different inputs are presented, the output channels (i.e., feature maps) should vary given they are meant to detect discriminative features[9].
An alternative method leverages network knowledge to analyze individual units' performance and relationships. This technique aims to understand network behavior and enhance its performance. Here's the core idea: the network is iteratively trained on a performance measure. Then, a relevance measure is calculated to identify the input or hidden units that contribute most significantly to performance. Finally, the least relevant units are automatically removed. This process, called skeletonization, simplifies networks by eliminating redundant units.[12] It should be noted that in this method there is a need to train the network with specific performance criteria, while our goal is to use pre-trained models that have been trained without specific performance criteria, or even their data in are not available.
Contrary to the research done, all our efforts are extracting the sub-network for fine-tuning and transferring general learning for use in specific fields, as well as having several small but learned models with high accuracy.Also, in this method, there is no need to use optimizers and complex loss functions, but it is implemented in the simplest form of this method.
It should be noted that most of the previous studies are in the field of deep learning and convolutional neural networks, which we will refer to in future studies.[18]

3. NNS : How to work

In this section, we discuss architecture, dataset and model training, And in the next section we will discuss the results

3.1. approach

The adopted method will work on the weights of the learned layers in such a way that if we intend to externalize the λ label prediction subnet:

- prepare data as classified input data related to λ category
- Monitoring neural network for each iteration and the effect frequency of neurons (Fig. 2)

  For each input, a snapshot of the neural layer is taken to identify common active neurons for common inputs (inputs with the same label).
- Zeroing unused or underused (low frequency) neurons for λ label prediction (Fig. 5)
- Storing the weights of the main model subnets
- Integrating the weights of each subnet
- storing the lightweight main model (3.5)

In order to zero out unused or underused neurons (low frequency), the average number of neuron calls is compared with the ζ threshold limit (hyperparameter) (eq. 1)(Fig. 4).

- x = The frequency of each neuron
- β = Max(Average frequency of neurons in the layer)

  For each node, the average active weights are calculated (Fig. 3).

$$f(x) = \begin{cases} x, & x \geq \zeta * \beta \\ 0, & otherwise \end{cases} \quad (1)$$

3.2. architecture

This architecture is the simplest multi-layer perceptron (MLP) architecture, which has a data input layer, a hidden layer, and an output layer, whose outputs are determined by (eq. 2)

$$y = f(\sum_{i=1}^{D} w_i x_i + b) \tag{2}$$

and the last layer will have softmax output (eq. 3)

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{3}$$

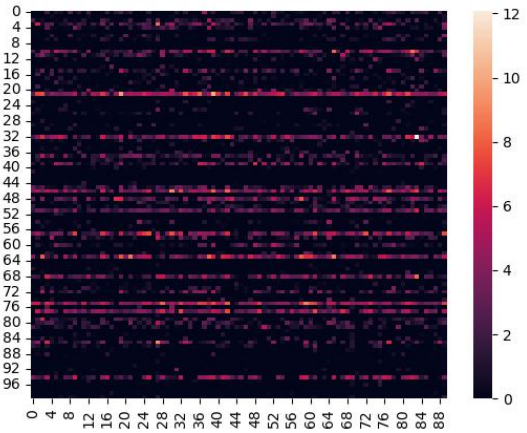

Figure 2: Each column is a snapshot of the layer per input to monitor active weights.

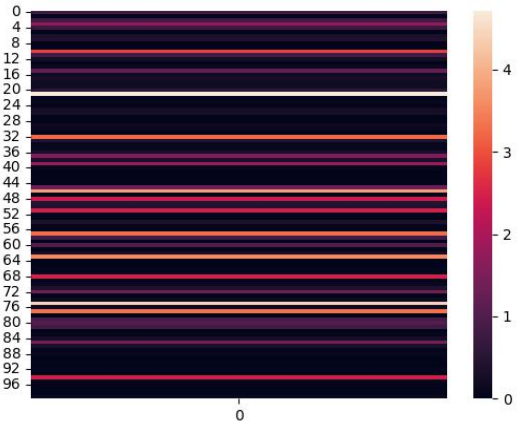

Figure 3: This image has one column that represents the layer and each row represents the neuron in that layer.

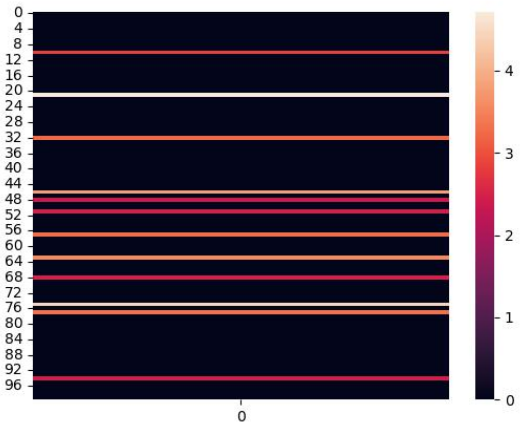

Figure 4: This image has one column that represents the layer and each row represents the neuron in that layer that obtained after filtering the values.

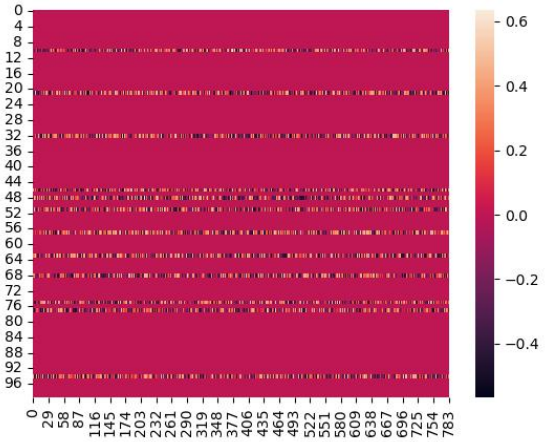

Figure 5: After identifying the active nodes to recognize the λ label, we set all nodes except the active nodes to zero in the weight matrix of the layer.

3.3. dataset

The MNIST database (Modified National Institute of Standards and Technology database) is a large collection of handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples (Fig. 6) [3].
Furthermore, the black and white images from NIST [14] were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.
It is a good 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.

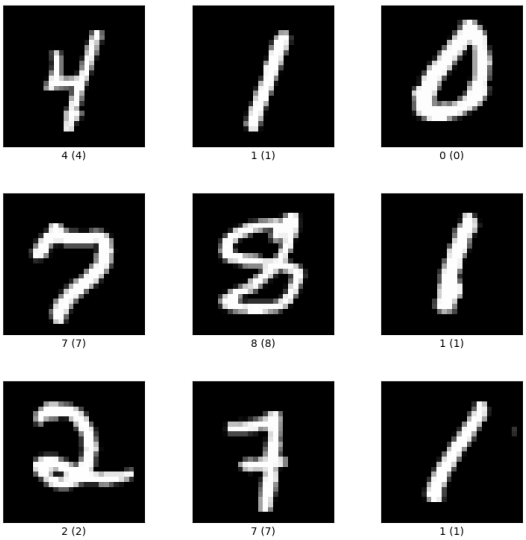

Figure 6: Sample data from MNIST dataset.

3.4. model training

This explanation is about a simple trained model [1]that uses the MNIST dataset, so more details can be provided regarding training the model and using its results to compare with NNS.

- activation functions : ReLU [1], SofMax [2]
- w1 Matrix = 100 * 28 * 28 * 1
- w2 Matrix = 10 * 100
- epoch = 200
- learning rate = 0.10
- Simple Model accuracy = 0.84
- individual labels accuracy (Table. 1)

3.5. Vector transformations

To save the weights in both stages :

- Storing the lightweight of the main model subnets (individual labels)
- Storing the lightweight main model

After filtering, the weight matrix is stored as a vector, which reduces the dimensions of the model in such a way that instead of storing a part of the weights, it generates them when the weights are loaded.

[1]code : https://github.com/saeidEmadi/NNS/blob/main/SimpleMLP.ipynb. Results and simple model weights are available in the GitHub folder [https://github.com/saeidEmadi/NNS] **/res** : *results* , **/wgts** : *weights*

Table 1: Accuracy of each label (simple MLP : accuracy of individual labels before using NNS)

label	accuracy (%)
0	94.3
1	92.7
2	79.8
3	88.9
4	78.4
5	71.9
6	87.6
7	87.6
8	73.1
9	79.3

Table 2: The file size of the models based on simple and vectorized for the ζ value of 0.4 for all labels (these values may be different for you (the weights file is attached next to the codes))

File Size (KB)		
label	**vectorized**	**routine**
0	684	1692
1	274	1519
2	245	1507
3	391	1569
4	391	1569
5	157	1470
6	450	1594
7	304	1532
8	362	1556
9	128	1458

Fig. 7 shows the vectorization of the weights after filtering (ignoring inactive nodes) and Fig. 8 shows the transformation of the vector into a matrix of weights (updating the values of the zero matrix to the dimensions of the number of neurons in the layer).
furthermore Tables 2 and 3 contain the values of the file size of models weights, which show the effect of vectoring on the size of the files. Also, Table 3 shows how much the size of the model file has changed before and after NNS implementation and has caused its size to decrease, it is obvious that the accuracy of the model has decreased somewhat, but it is a value that can be ignored to some extent (accuracy are attached in the results section). This operation is also used to extract individual model weights for separate labels (only that label).

Table 3: File size of model weights (simple model and with NNS implemented on it) based on simple and vector for ζ value of 0.4 .

File Size (KB)		
	vectorized	**routine**
Original weights	-	2658
Integrated weights (NNS)	**684**	1692

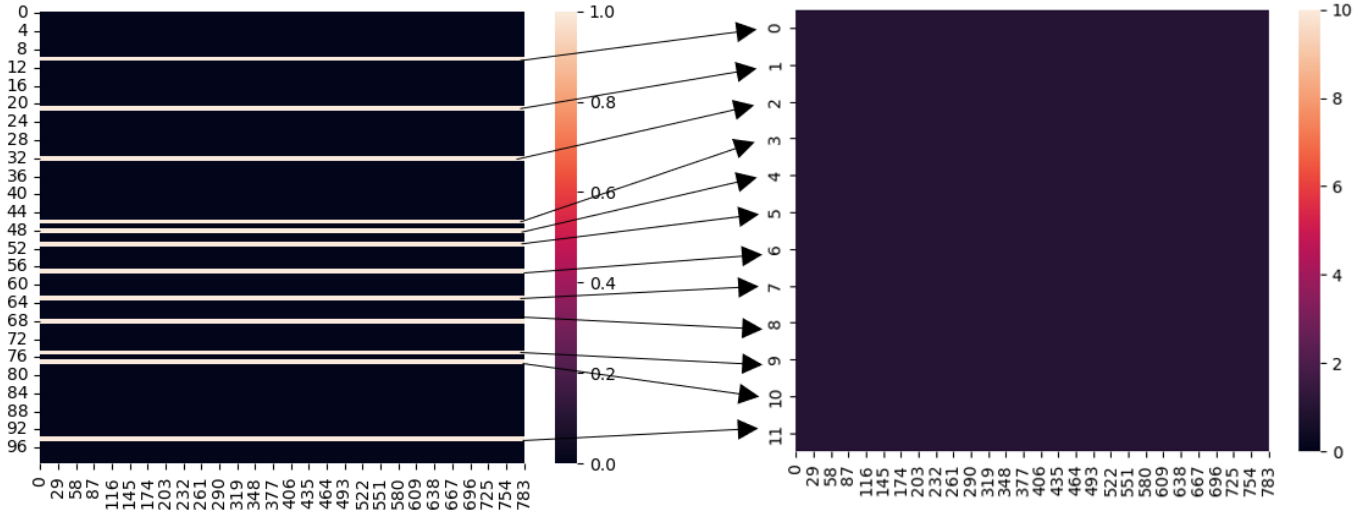

Figure 7: Convert the matrix to a meaningful vector to reduce the file size of the weight model.

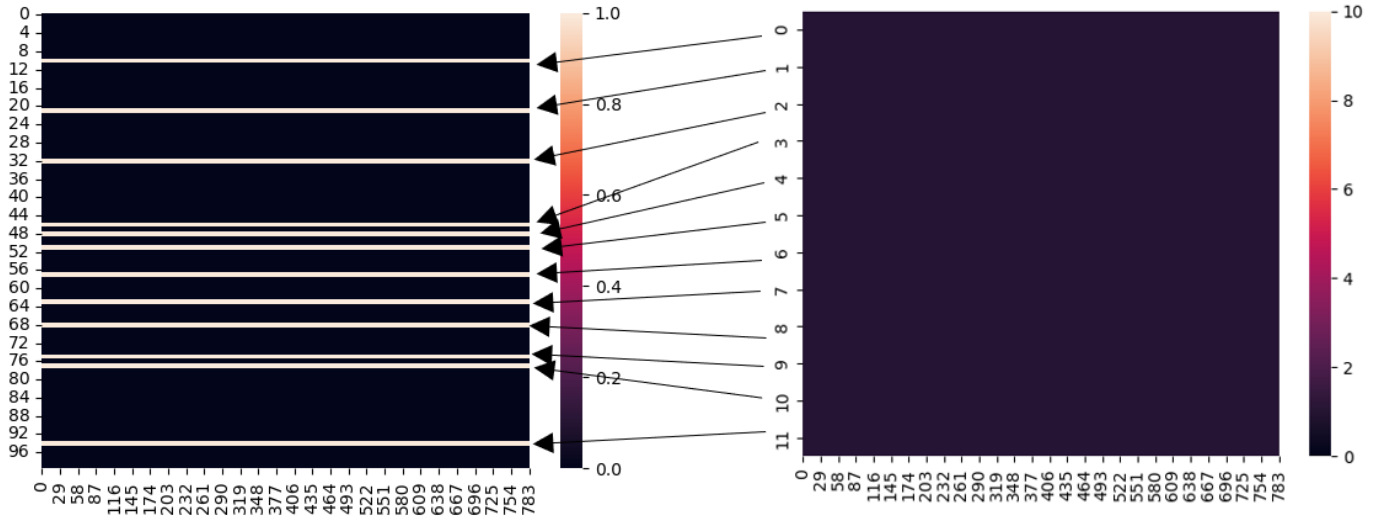

Figure 8: Converting the weighted vector to the matrix of weights (first the zero matrix is created and then the values mentioned in the vector are updated).

4. Results and Discussion

After all the steps are done, the results provide significant data to compare between the simple model and the filtered model (NNS). The attached results are shown in Tables 4 and 5 for ζ values of 0.4 and 0.5 . Figure 4. It shows the results obtained from the step-by-step implementation of nns and its effect on the accuracy of other labels. Also, the accuracy of the label can be compared with the simple model and nns. Table 4 shows the step-by-step addition of tag recognition features in the main model, each column shows the results of adding that column and our columns before it. By carefully looking at this table, it can be understood that by adding the predictive features of ***label 1***, the model also finds significant accuracy for predicting ***label 9***, which indicates that their features are in common, or the columns of ***label 4*** totally shows that NNS caused the disintegration of its predictive features and its accuracy did not reach above **10%**. The plots in Figure 9 also show the effect of adding other labels on accuracy.
Table 5 shows the number of parameters and their accuracy for both ζ, **It is very important to compare the number of parameters in the original model layer (28*28*100) with the parameters used to detect the desired label,** ***and the significant difference in the number of parameters as well as the small change in accuracy can be considered a very good point for NNS.***
It is clear from the obtained results that **the accuracy of the NNS output is higher than the accuracy of the simple model in the detection of some tags** (such as tag 9). Of course, in some labels (except for label 4), the accuracy has decreased, but due to the law difference, they can be ignored.

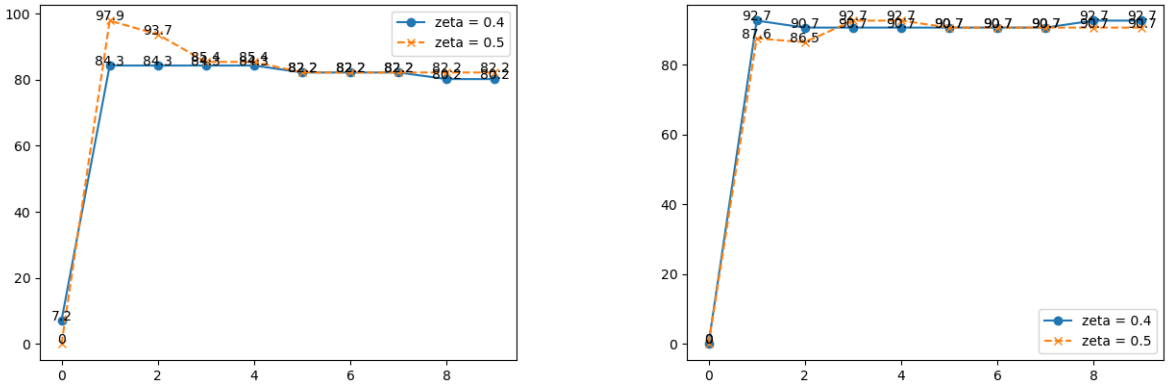

Figure 9: This plot shows the effect of adding one-to-one tag recognition features on the recognition accuracy of tags 9 and 1. (for ζ 0.4 and 0.5) (right figure for label 9 and left figure for label 1).

However, considering the accuracy (high increase and low decrease) as well as the size of the model parameters (low parameters reduced the size of the model), we can conclude that the pruned model is a better choice. Also, its accuracy and size is better and smaller than the simple model.
some the special results of this research is:

- Extracting specific features of a comprehensive model for specific label prediction
- Extracting the features of the comprehensive model and reducing the dimensions of the model with the highest accuracy, or in other words, compressing heavy and large models into small and light models.

It should be noted that in the article only values of 0.5 and 0.4 have been considered for ζ, it is obvious that this value can be reduced or increased according to the desired accuracy, that is why this parameter is considered as a ***hyperparameter***.

5. Conclusion

There has always been a question whether very large, heavy and comprehensive models can be used for smaller and lighter devices or not, and also what ratio of the accuracy of the extracted models will be to the original accuracy. **This research and its results showed well that it is possible to extract a sub-network of the original model by simply considering the active nodes and finding the predictive features of a specific label without the need for primary training resources and data predicts that particular label.** NNS provides the possibility to prune the original models and transform them into lighter models or extract from it a subnet that is suitable for running on smaller and lighter devices and resources.

Of course, it should be noted that this paper and research has more research capacity because it has used the subnets obtained from the original model to predict only a subset of the set of labels of the original model. In the research of this article, this research was also done, but the correct criteria for predicting that label was not found in multi-models.
In the next steps, to continue this research, you can go on NNS implementation for :

- CNNs [15]
- RNNs [17]
- especially large language models (LLMs) [5]
- Correct validation criterion of extracted sub-network for multi-models

Because the said models have 1x Mylonian parameters, which makes them heavy, and they can be made much lighter (of course, with the desired accuracy and correct setting of the ζ value).

Table 4: The values contain accuracy the results obtained during the step-by-step execution of NNS on the original model. The labels in the rows represent the model labels and each column represents the label that NNS is running at that stage (Each column is cumulatively, for example, column 4 (label 1) shows that the obtained integrated model contains the unique features of column 4 and the ones before it (labels 0 and label 1) and also what effect It has the accuracy of recognizing labels (each row)). It is clear that with the addition of new features (adding the next column), the accuracy of recognizing labels (rows) improves in some labels (eg. labels 9) or loses accuracy in some labels (eg. labels 3)

		accuracy (%)										
zeta	**label**	***0***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***routine***
0.4	**0**	**100**	93.4	82.2	82.2	82.2	80.3	83.1	83.1	81.3	81.3	94.4
0.5		**100**	98.1	77.5	77.5	77.5	80	83.1	83.1	83.1	83.1	
0.4	**1**	7.2	**84.3**	**84.3**	**84.3**	**84.3**	82.2	82.2	82.2	80.2	80.2	92.7
0.5		0	**97.9**	93.7	85.4	85.4	82.2	82.2	82.2	82.2	82.2	
0.4	**2**	1.6	0	79.8	79.8	79.8	80.6	79.8	79.8	**81.4**	**81.4**	79.8
0.5		0	0	**87**	75	75	80	79.8	79.8	79.8	79.8	
0.4	**3**	4.4	57.7	87.7	87.7	87.7	**88.8**	85.5	85.5	83.3	83.3	88.8
0.5		4.4	8.8	52.2	87.7	87.7	**88.8**	85.5	85.5	85.5	85.5	
0.4	**4**	0	2.9	0.9	0.9	0.9	1.9	2.9	2.9	**3.9**	**3.9**	78.4
0.5		0	0	0	0	0	1.9	**2.9**	**2.9**	**2.9**	**2.9**	
0.4	**5**	2.2	44.9	51.6	51.6	51.6	58.4	**68.5**	**68.5**	67.4	67.4	74.9
0.5		2.2	30.3	43.8	51.6	51.6	58.4	**68.5**	**68.5**	**68.5**	**68.5**	
0.4	**6**	1	9.2	61.8	61.8	61.8	64.9	**75.2**	**75.2**	72.1	72.1	87.6
0.5		13.4	4.1	19.5	58.7	58.7	64.9	**75.2**	**75.2**	**75.2**	**75.2**	
0.4	**7**	26.6	43.8	55.2	55.2	55.2	**59**	**59**	**59**	55.2	55.2	87.6
0.5		8.5	18	17.1	52.3	52.3	**59**	**59**	**59**	**59**	**59**	
0.4	**8**	0	65.5	61.2	61.2	61.2	61.2	59.1	59.1	**75.2**	**75.2**	73.1
0.5		0	29	20.4	51.1	51.6	**61.2**	59.1	59.1	59.1	59.1	
0.5	**9**	0	**92.7**	90.7	90.7	90.7	90.7	90.7	90.7	**92.7**	**92.7**	79.3
0.4		0	87.6	86.5	**92.7**	**92.7**	90.7	90.7	90.7	90.7	90.7	

Table 5: This result shows the number of parameters and their accuracy for both zeta, it is important to compare the number of parameters in the original model layer (28x28x100) with the parameters used to detect the desired label (each row). The difference in the number of parameters and accuracy is very impressive and important (reduction of parameter frequency from (28*28*100) to (28*28*5) (5 here means that only 5 features (nodes) are needed to recognize label 1) and increase of accuracy from 92.7 to 98.9 in label 1 (This increase in accuracy shows the extraordinary power of this approach) and ζ 0.5)

	number of parameters		**accuracy (%)**		
label	**NNS**				**routine**
Zeta	*0.4*	*0.5*	*0.4*	*0.5*	-
0	6272	3136	100	100	94.4
1	5488	3920	88.5	98.9	92.7
2	9408	6272	91.9	94.3	79.8
3	9408	8624	96.6	100	88.8
4	3136	1568	0	0	78.4
5	10976	9408	55.0	88.7	74.9
6	7056	5488	93.8	85.5	87.6
7	2352	1568	56.2	77.1	87.6
8	8624	4704	77.4	77.4	73.1
9	2352	2352	100	100	79.3

References

[1] A. F. Agarap. Deep learning using rectified linear units (relu), 2019.

[2] K. Banerjee, V. P. C, R. R. Gupta, K. Vyas, A. H, and B. Mishra. Exploring alternatives to softmax function, 2020.

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

[4] Z. Fu, H. Yang, A. M.-C. So, W. Lam, L. Bing, and N. Collier. On the effectiveness of parameter-efficient fine-tuning, 2022.

[5] T. Gao, H. Yen, J. Yu, and D. Chen. Enabling large language models to generate text with citations, 2023.

[6] E. Grossi and M. Buscema. Introduction to artificial neural networks. *European journal of gastroenterology & hepatology*, 19 12:1046–54, 2007.

[7] G. Kumar and P. K. Bhatia. A detailed review of feature extraction in image processing systems. In *2014 Fourth International Conference on Advanced Computing Communication Technologies*, pages 5–12, 2014.

[8] 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.

[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

[10] J.-H. Luo and J. Wu. An entropy-based pruning method for cnn compression, 2017.

[11] M. McGill and P. Perona. Deciding how to decide: Dynamic routing in artificial neural networks, 2017.

[12] M. C. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[13] D.-K. Nguyen and T. Okatani. Multi-task learning of hierarchical vision-language representation, 2018.

[14] N. I. of Standards and Technology. Security requirements for cryptographic modules. Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 140-2, Change Notice 2 December 03, 2002, U.S. Department of Commerce, Washington, D.C., 2001.

[15] K. O'Shea and R. Nash. An introduction to convolutional neural networks, 2015.

[16] L. Pinto and A. Gupta. Learning to push by grasping: Using multiple tasks for effective learning, 2016.

[17] B. Quast. rnn: a recurrent neural network in r. *Working Papers*, 2016.

[18] S. Vadera and S. Ameen. Methods for pruning deep neural networks, 2021.

[19] Y. Wang, X. Hu, and H. Su. Interpretable disentanglement of neural networks by extracting class-specific subnetwork, 2019.

[20] X. Yin, W. Chen, X. Wu, and H. Yue. Fine-tuning and visualization of convolutional neural networks. In *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1310–1315, 2017.

[21] Y. Zhang and Q. Yang. A survey on multi-task learning, 2021.

[22] Z. Zhao, L. Alzubaidi, J. Zhang, Y. Duan, and Y. Gu. A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations. *Expert Systems with Applications*, 242:122807, 2024.

[23] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning, 2020.