Embarrassingly Simple One Shot Learning and Siamese Neural Networks Based on Deep Features

Anonymous Author(s)
Affiliation
Address
email

Abstract

One shot learning has caught much attention in recent years, we are inspiring to take a look at this area and try something by ourselves. In this paper, we realized several image features extractors and compared their performances. Then we focus on a classic zero shot learning algorithm, and made efforts to change its structure a little bit for the purpose of applying it to one shot tasks. We also tried dimensionality elimination methods to get our extracted deep features more neat. In the experiment process, we make a lot comparisons to find out the most suitable deep features extractor and dimension reducer for our models. Finally, we chose the best deep features extractor and dimensionality reduction method to train Siamese Neural Networks. Most of our models give better performances than the baseline fine-tune.

1 Introduction

With the rapid progress of machine learning technologies, nowadays, we have number of algorithms to make computers learn things the way we human beings do. Especially, deep convolutional neural networks [8][15][6] have achieved encouraging results on image classification. For example, in ILSVR (ImageNet Large Scale Visual Recognition Competition), which is held by ImageNet from 2010 to 2017, deep learning algorithms have reached impressive results better than that human beings can get (**Figure 1**). However, classical deep learning algorithms are data hungry, i.e they need a huge amount of labelled datasets which costs too much time to label and is commonly impossible to get in real situations.

In contrast, the human visual system exhibits the excellent ability to learn new things from only one or a few examples. For a human brain, it has both the ability to memorize and the ability to find out deep features from the original vision. For instance, a 5 years old child born in Asia may have never seen a lion, while he can tell what is a tiger. Though the lion is a novel class for him to recognize, he can tell that this alien 'big cat' has long and sharp tails, just as the tiger, so he can easily draw the conclusion that this animal is dangerous. Moreover, the next time he saw a lion, he would be able to realize the fact that he had seen this animal before. These abilities are mainly based on long and short memory and the ability to summarize the similar features.

Inspired by our human brains, zero shot learning and few shot learning have caught increasingly attentions in recent years. In one shot learning [10], we have diverse ways to add prior knowledges to the model, such as Bayesian [10], features [7] and semantic attributes [20], and then train the model using only one or few examples from each novel class. For the classifier, though we also have many other methods such as kNN, SVR and Logistic Regression, mostly we use convolutional neural networks, the reason will be discussed in Section 4.

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

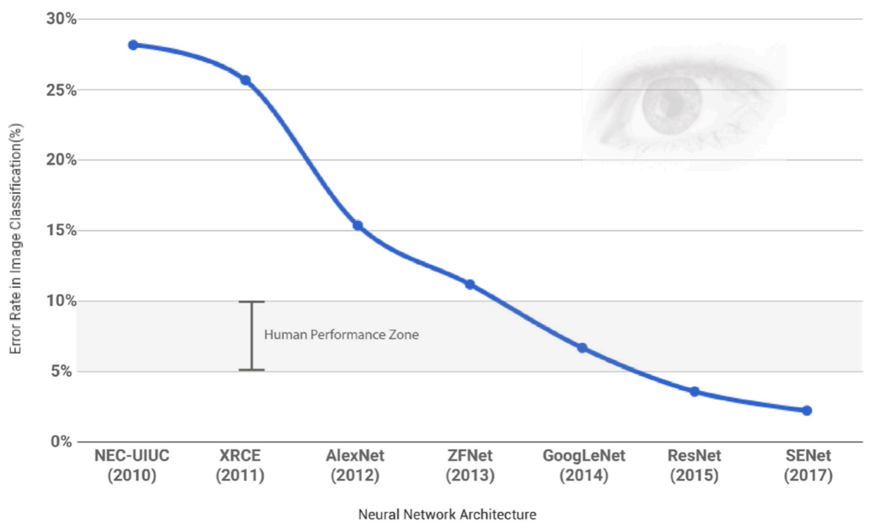

Figure 1: Development performances in ILSVR

In this paper, we focus on one shot learning, and also introduce the way to apply an zero shot learning algorithm on one shot tasks. Specifically, we solve an one/few shot task for image classification on mini-imagenet. The experimental protocol follows Section 2 in [12], which means we divide 100 classes to 2 parts, 80 for pre-training and 20 for testing. In each parts, the dataset D_{meta} has a split of D_{train} and D_{test}. There are 600 pictures in each class, and in pre-training set, we take 550 as D_{train} and 50 as $D_{validation}$.

Generally, our work can be summarized into 3 parts:

1. **Realizing several image features extractors and comparing their performances**. We select some classic ConNet as deep features extractors, including AlexNet [9], VGG16 [16], ResNet50 [6] and DenseNet121 [3]. Firstly, we compare their classification performance on validation set. Then we extract parameters from the last fully connected layers as the deep features. Finally, using these features as inputs, compare the performances on linear classification.
2. **Applying Embarrassingly Simple Zero Shot Learning method (ESZSL, [13]) to one shot tasks**. ESZSL is originally raised to tackle with zero shot learning tasks. It has an extremely simple structure but performing very well on zero shot learning. First of all, we managed to apply it on one shot learning, which we called the Embarrassingly Simple One Shot Learning (ESOSL). Secondly, we use PAC and LLE to reduce the dimension of features and see how it influences the results. Then training the ESOSL with features generated by ResNet50 and DenseNet121. Finally, comparing the performances and find out the best features extractor.
3. **Construct Siamese Neural Networks**. We train and evaluate Siamese Neural Networks using DenseNet121 as the deep features extractor.

2 Related Work

For convolutional neural networks used in one shot learning, basically we have several different types: **(a)** Based on metric. These models are all based on metric computation to do classification, however they use different kinds of metric, different kinds of features and have different structure. Siamese neural networks [2] which compares the similarity between a training example of a novel category and a test example, matching networks [19] which employ a differentiable nearest neighbor classifer implemented with an attention mechanisim, and prototypical networks [17], which takes an average of the feature vectors extracted from training examples as prototype feature vector for learing, can be put into this type. **(b)** Based on graph neural net work. In [4], the author established a deep GNN to realize one shot learning. **(c)** Based on meta learning. Meta learning, which is also called learning to learn [18], typically involve a meta-learner model which tries to learn a learner model for a new task within a few shots [1]. Thus we have memory-augmented neural networks [14] and meta-learning LSTM [12]. In a nutshell, a ConNet model can own the properties of more than one type at the same time, and these types offer us inspirations from different angles.

Embarrassingly Simple Zero Shot Learning method (ESZSL) is developed by Romera and Philip [13] in 2015. This method is based on a general framework which models the relationship between features and classes as a two linear layers network, with top layer weights directly given by the environment. Owing to its assumptions and special regularizer, the algorithm can be very neat and short, however, proved to perform well on zero shot learning. Thanks to its simple structure, we can easily implement it to work on one shot learning, which we called the Embarrassingly Simple One Shot Learning (ESOSL).

Both convolutional neural networks and ESOSL require deep features or attributes as inputs. Neural networks have been introduced to extract features from the origin vision [11][7]. In this paper, we use several famous neural networks, which perform well in ILSVR, delete the output layer and get the output of the last fully connected layer as our deep features.

3 Methodology

3.1 Image Features Extractors

AlexNet [9] This network is the champion for ILSVR in 2012, but it has much bigger contribution that AlexNet is the first model tried on ILSVR which is based on neural networks. Besides, it is the first neural network model using ReLu, applying Dropout and max pooling. It has only 8 layers, and all layers except the first convolution layer are fully connected (**Figure 2** [1]).

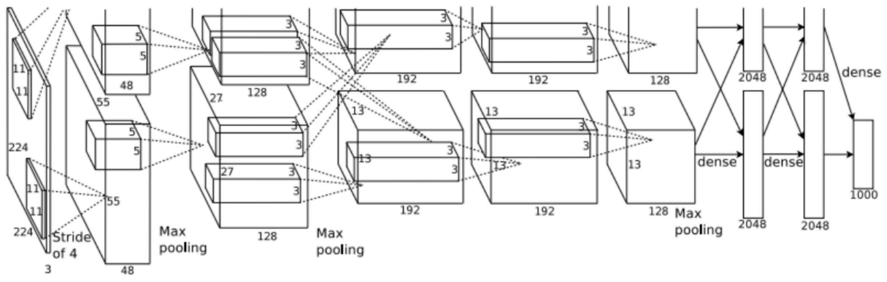

Figure 2: The structure of AlexNet.

VGG [16] VGG is the second place for ILSVR in 2014. It is designed to study the influence of the depth. Following the basic idea of AlexNet, VGG also applies the CONV-POOL-FC structure, but has smaller convolution kernel to deepen the network (**Figure 3** [2]).

ResNet [6] ResNet is the champion for ILSVR in 2015, and it created a new structure called residual blocks. Residual structure enables each layer propagates information not only to the next layer but also further layers (**Figure 4**). It is helpful to avoid gradient elimination and effectively increase the generalization ability of neural network. We apply ResNet50 in this paper, which has 50 layers.

DenseNet [3] DenseNet is the best paper for CVPR 2017. Though ResNet has used residual to widen the input of hidden layers, it is not completely a wide ConNet. Different from ResNet which applys residual propagation between every several layers and ignores residuals in backward propagation, DenseNet is constructed by dense blocks and transition layers, and in each dense block

[1]Available at `http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

[2]Available at `http://www.cs.toronto.edu/~frossard/post/vgg16/`

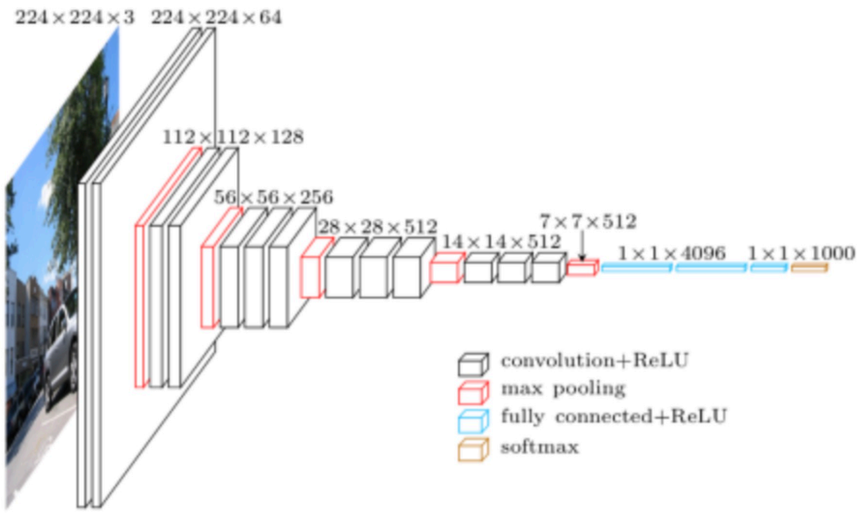

Figure 3: The structure of VGG16.

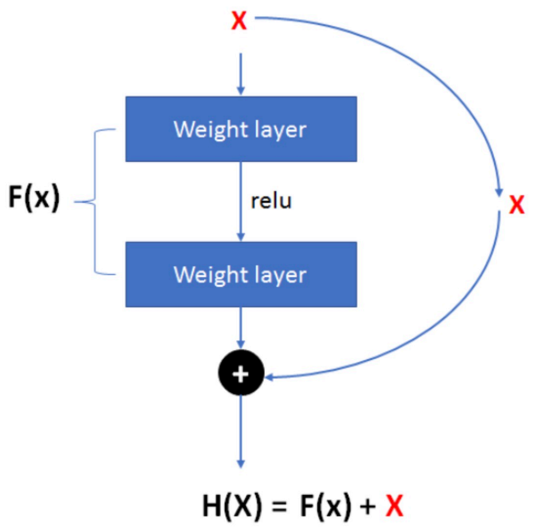

Figure 4: The structure of a residual block.

every layer propagates its output to all the layers behind it (**Figure 5** [3]). As a result, DenseNet can be much deeper than ResNet. We use DenseNet121, which has 121 layers, in this paper.

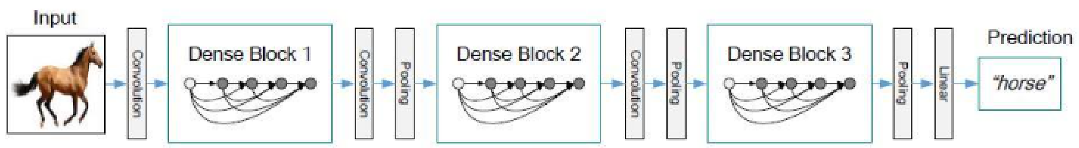

Figure 5: The structure of DenseNet.

Aforementioned models are originally designed for classification task with a big load of training set. In this paper, we use these neural networks as deep features extractor, which means there is no need to remain the last classification layer. So we just remove the last layer of these models, and

[3]Available at `https://www.cnblogs.com/skyfsm/p/8451834.html`

employ the new output as features. These features may not be interpretable, thus they are called deep features.

3.2 Embarrassingly Simple learning

Before we start, let's define some notations first. Assume there are z classes at training stage, each of them having a features, and we denote the feature matrix as $S \in [0,1]^{a*z}$. Let us denote by $X \in R^{d*m}$ the training examples, where d and m are the dimensionality of the data and the number of instances. Denote the truth labels of training set by Y.

When we use a linear classifer, e.g. logistic regression, we would optimize the following problem:

$$minimize L(X^T W, Y) + \Omega(W) \tag{1}$$

Here we would like to consider the impact of features, so we rewrite weights as$W = VS^T$:

$$minimize L(X^T V S^T, Y) + \Omega(V) \tag{2}$$

To simplify the calculation, we design a regularizer as the following form:

$$\Omega(V; S, X) = \gamma ||VS||^2_{Fro} + \lambda ||X^T V||^2_{Fro} + \beta ||V||^2_{Fro} \tag{3}$$

where the scalars γ, λ, β are the hyper-parameters of this regularizer, and $|| \ ||_{Fro}$ denotes the Frobenius norm.

Furtherly, if we have $L(P, Y) = ||P - Y||^2_{Fro}$ and $\beta = \gamma\lambda$, then the solution to the question can be expressed in closed form:

$$V = (XX^T + \gamma I)^{-1} XYS^T (SS^T + \lambda I)^{-1} \tag{4}$$

This ZSL algorithm is offered by Romera and Philip in [13], and now we managed to implement it to do OSL. The key is to compute matrix V using features extracted from pre-training set and set the feature matrix on test-training set as S. Then we can easily get the weight matrix $W = VS^T$ and use it in the linear model to do prediction. As mentioned in Section4, logistic regression gave the best performance among 3 linear classification models, so we use logistic regression as the classifer for our ESOSL model.

Moreover, we generate 1000 features for each picture, and in fact most feature matrices are sparse. So we also use Principle Components Analysis (PCA) and Locally Linear Embedding (LLE) to simplify our features.

3.3 Siamese Neural Network

Siamese Neural Network is metric-based which employs a unique structure to judge similarity between inputs and examples. It consists of a pair of twin networks which accept distinct inputs and are jointed by an energy function. This energy function is used to quantify the similarity of deep representations of the twin networks by computing a certain metric. As showed in **Figure 6**, the net work also has several layers after the joint of two channels. To optimize the net, we use a unique loss function which called the contrasive loss:

$$L = \frac{1}{2N} \sum_{i=1}^{N} y_i d_i^2 + (1 - y_i) mac(margin - d_i, 0)^2 \tag{5}$$

4 Experimental Results

4.1 Deep Features Extraction

4.1.1 Image Features Extractors Classification

We train AlexNet, VGG16, ResNet50 and DenseNet121 and evaluate them on pre-validation set, and get their top-1 and top-5 accuracy in **Table 1**:

From **Tabel 1** we noticed that DenseNet121 performs best for both top-1 and top-5 accuracies.

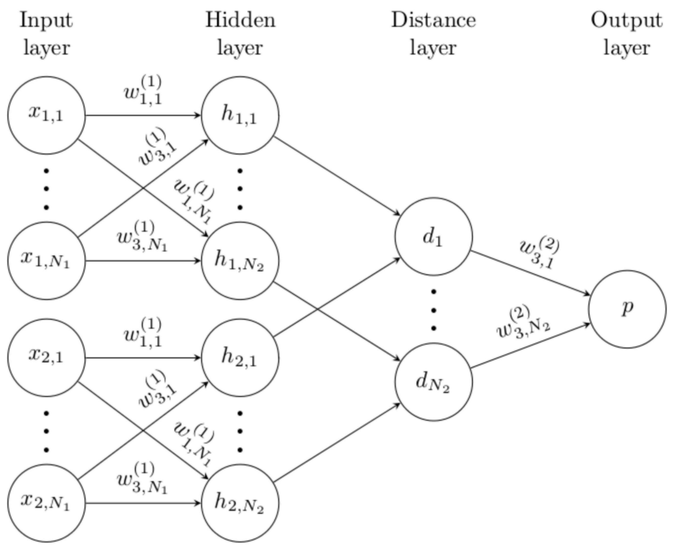

Figure 6: The structure of SiameseNN.

Table 1: Top-1 and top-5 accuracies on pre-validation set

	Accuracy	
Model	Top 1	Top 5
Alex	40.2	70.7
VGG16	39.0	68.9
ResNet50	43.6	73.0
DenseNet121	**47.7**	**76.5**

4.1.2 Linear Classification With Diverse Extractors

Now let's use the aforementioned convolutional neural networks to extract deep features on mini-Imagenet. We apply different features extractors on SVR, kNN ($k = 3$) and Logistic Regression models with all the combinations of $k = 1$ or $k = 5$ and $N = 5$ or $N = 20$, where k is the number of shot and N is number of noble classes tested which is also called the 'way'.

Table 2: Comparison of different extractors applying on linear models

Extractors	N=5						N=20					
	k=1			k=5			k=1			k=5		
	SVR	kNN	LR	SVR	kNN	LR	SVR	kNN	LR	SVR	kNN	LR
AlexNet	22.7	20.5	45.3	29.0	25.5	60.2	8.5	6.5	21.7	11.3	11.4	32.2
VGG16	22.4	20.7	37.7	27.7	25.6	52.0	8.2	6.3	19.6	11.4	11.4	31.8
ResNet50	**39.7**	**22.8**	**57.5**	**57.2**	**52.3**	**72.4**	14.3	**10.4**	21.5	22.4	19.0	32.9
DenseNet121	33.0	22.6	50.2	46.3	42.6	66.0	**14.5**	9.5	**22.6**	**22.5**	**19.7**	**35.2**

From **Table 2**, we find that ResNet50 and DenseNet121 have much better performances on all situations. This is easy to understand for the reason that ResNet50 and DenseNet121 have much deeper nets than the other two, so they are supposed to own better generalization ability on complex image sets. More detailed, though DenseNet121 performs better on pre-validation classification task than ResNet50 as shown in **Table 1**, it doesn't indicate that it can always get more useful deep features as shown in **Table 2**. Otherwise, since ResNet50 and DenseNet121 are far better than the other two, we only use ResNet50 and DenseNet121 in the following experiments .

4.2 ESOSL

4.2.1 PCA and LLE for ESOSL

Our extractors return 1000 deep features for each picture, however, most features matrices are spare. As a result, we naturally consider dimensionality reduction such as PCA and LLE. Firstly we eliminate the dimensionality of deep features, and then set these remaining features as inputs to run our ESOSL model. We tried several distinct values of remained deep features of both PCA and LLE to see how it effects our results. As all experiments taken under $N = 5$ condition, the results are shown as **Figure 7**.

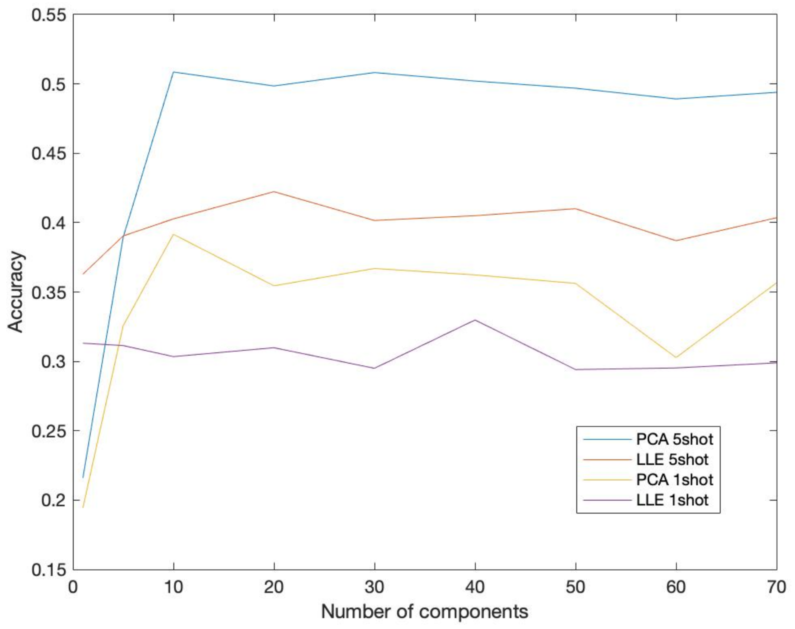

Figure 7: PCA and LLE performances for ESOSL with different number of components remained

As shown in the **Figure 7**, PCA works much better than LLE under both conditions. Besides, the performances tend to be stable after the number of remaining components is more than 10. Therefore, to simplify the computation, we will use PCA with remained components as 30 to reduce dimension first to have our deep features more neat. On the other hand, those unremained features can be considered to be noise, which implies that remained deep features are likely to have more powerful discriminative properties than original ones.

4.2.2 ResNet50 and DenseNet121 for ESOSL

Now after applying PCA, we make a comparison between ResNet50 and DenseNet121 again. We set reduced deep features extracted from the two models as ESOSL inputs respectively, then train and evaluate ESOSL model.

Table 3: ESOSL performances using Different Features Extractors

Extractors	N=5		N=20	
	k=1	k=5	k=1	k=5
ResNet50	39.7	54.9	18.2	29.1
DenseNet121	**44.0**	**61.0**	**20.5**	**30.3**

Results are shown in **Table 3**. We can safely draw a conclusion that DenseNet121 returns a better performances on all situations. So in the following experiments, we'd like to use DenseNet121 as our image feature extractor.

4.3 SiameseNN

To train the SiameseNN, we are supposed to generate random pairs first. Within pre-training set, there are 80 classes with 550 pre-train examples per class. We managed to generate pairs with each image appearing in pairs at least twice, so we get 88000 pairs in total. According to average euclidean distance between different classes, we set the margin parameter in the contrastive loss as 10. In the **Table 4**, we give results of SiameseNN with that of linear models and ESOSL, and compare with the result of a general finetune as a baseline. Plus, we show at the last row the performances of one of the most state-of-art models [5].

Table 4: Performances of All Models on Mini-Imagenet with Different k-shot and N-way

Models	N=5		N=20	
	k=1	k=5	k=1	k=5
Baseline finetune	28.09	49.8		
kNN	22.6	42.6	9.5	19.7
SVR	33.0	46.3	14.5	22.5
LR	**50.2**	**66.0**	22.6	35.2
ESOSL	44.0	61.0	20.5	30.3
SiameseNN	41.7	53.0	**32.2**	**39.1**
CosineNN	*56.2*	*73.0*		

Our models LR, ESOSL and SiameseNN based on deep features perform better than the baseline. For 5-way learning, simple linear regression performs better; While for 20-way learning, SiameseNN is more accurate. This result is within our expectation, cause 5-way learning is not a very complex task and doesn't need a complex nonlinear model to represent, while 20-way is much more complex, so Neural Networks can have better performances owing to their outstanding generalization ability. There is a **Figure 8** which could be a good proof for our analysis.

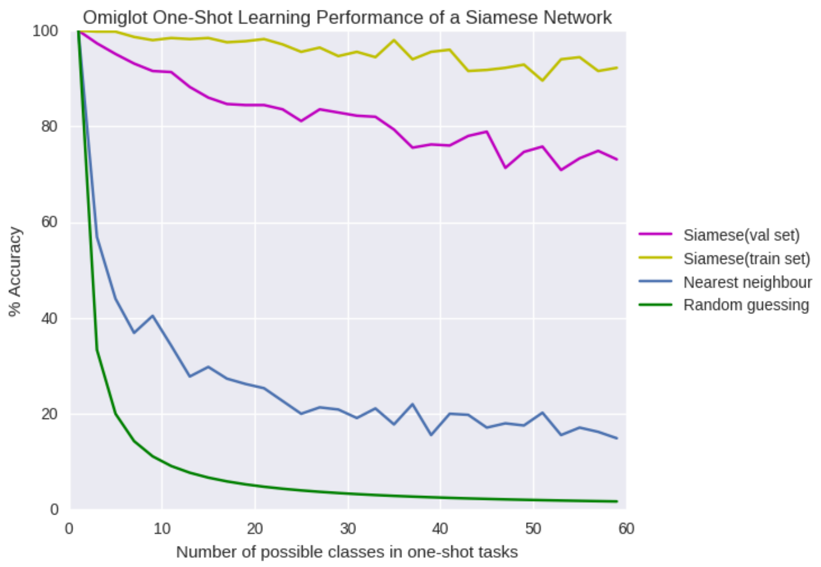

Figure 8: Performances of different models with the number of noble classes changing.

5 Conclusion

In a nutshell, we firstly realized several image features extractors including AlexNet, VGG16, ResNet50 and DenseNet121, and comparing their performances; then we applied ESZSL method to do one-shot task and used PCA and LLE to reduce the dimensionality of deep features; finally, we trained Siamese Neural Networks. We have done a lot comparisons, and found out the most suitable deep features extractors and dimensionality elimination method.

However, our work still has much space to improve. Restricted by time and device, data feature extractors are only trained for 5 epochs and SiameseNN is only trained for 10 epochs. We believe they may have better performance if trained more. Besides, we only use LR as classifer in ESOSL, and it will be quite interesting to try some other classifers.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. 2016.

[2] R Zemel G Koch and R Slakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep learning workshop*, 2015.

[3] Huang Gao, Liu Zhuang, and Kilian Q Weinberger. Densely connected convolutional networks. 2016.

[4] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. 2017.

[5] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. 2018.

[6] Ren S et al. He K, Zhang X. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[7] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall. Learning a kernel function for classification with small training samples. In *International Conference*, 2006.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems*, 2012.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems*, 2012.

[10] Rob Fergus Li Fei-Fei and Pietro Perona. One-shot learning of object categories. In *IEEE transactions on pattern analysis and machine intelligence*, 2006.

[11] . Mao, J. and A K Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, 2002.

[12] S. Ravi and H.Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[13] Bernardino Romera-Paredes and Philip H S Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on International Conference on Machine Learning*, 2015.

[14] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. 2016.

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014.

[17] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. 2017.

[18] S. Thrun. Learning to learn: Introduction. 1996.

[19] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. 2016.

[20] Fu Yanwei, Timothy M Hospedales, Xiang Tao, and Gong Shaogang. Learning multimodal latent attributes. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 36(2):303, 2014.