Title: Limited data analysis

Abstract

This project focuses on analyzing various methods such as meta learning, data augmentation, contrastive pre-training, and architecture parameters to solve the problem of limited data. Pre-training with contrastive learning can be useful in large batch regime with a good selection of augmentation but with smaller batch sizes and poor augmentations it can hurt model generalization instead of improving it. In our experiments we find that when dropout(with ratio 20%) was added to the baseline it decreased the classification accuracy by 6% showing that important features are learned by the baseline model which are not learnt well in presence of dropout in the limited data regime. When we used reptile algorithm it could find good initialisation for the network we find and that we get 17% test accuracy. Using prototypical typical networks, we get the best test accuracy (using episodic testing) of 25.26% when we use 4-way, 5 support and query for image size 224*224.

1. Introduction

Deep learning has become the most popular and fastest growing tool in Machine Learning (ML) and Deep Neural Networks (DNN) (Mikołajczyk & Grochowski, 2018). Deep convolutional neural networks perform well in a variety of computer vision applications, resulting in considerable advances. Convolutional neural networks (CNN) that maintain image spatial features have been created and effectively used to Computer Vision tasks as image classification, image segmentation, and object detection thanks to these networks. Because CNNs reduce image resolution, they perform better in Computer Vision tasks as they are lower-dimensional and more useful technique (Shorten & Khoshgoftaar, 2019).

Since its inception, Convolutional Neural Networks (CNN) have been employed in a variety of domains, including handwriting recognition, object identification, and speech analysis (Qiao & Ma, 2018), with successful results. They are easier to train than other neural network architectures since each convolution layer have relatively sparse connections, and they have crucial properties like reducing the size of the feature maps and boosting computation performance (Pang et al., 2017). Despite the fact that CNNs perform well in a variety of classification tasks, there are still some challenges to overcome since, as indicated in the literature, they sometimes require exceptionally big datasets to provide excellent results in a variety of tasks. However, in the real world, gathering large amounts of data for these tasks is commonly a significant cost or time constraint. Furthermore, achieving high success in performances is extremely difficult since the acquired data has additional issues, such as being unreliable and wrongly classified (Look & Riedelbauch, 2019). Many research have shown that bigger data can lead to more successful deep learning models (Halevy et al., 2009) (Sun et al., 2017). For example, the research in the studies conducted by (Oquab et al., 2014) and (Simonyan & Zisserman, 2014) show that large labelled datasets are essential for part of CNN's effectiveness.

We focus on the absence of adequate training data in this work, which is one of the most serious issues in machine learning. Methods for classifying utilising the ImageNet dataset, which has artificially decreased the amount of samples per class, are provided as a possible solution to this problem. The data augmentation approach, which allows to expand the current data variety without actually collecting extra data, is offered as a possible solution to overcome the shortage of information in classification problems, when the available data is insufficient to train the classifiers efficiently. Data augmentation allows for the creation of new samples by performing suitable data transformations without changing the labels of existing data (Fawzi et al., 2016). In addition, overfitting and generalisation capabilities are also issues with datasets with a limited quantity of data. The main advantage of data augmentation is that it reduces overfitting by enabling the model to better generalise the data when more training data is received (Fonseka & Chrysoulas, 2020). However, many data augmentation operations can introduce a data distribution bias that can make it difficult to take full advantage of the augmented data by changing the data distribution between the augmented data and the original data during training progress. In addition, it can be an exhausting process (Xu et al., 2020).

Another method used to overcome the limited data problem is the Few-Shot learning approach. Few-Shot learning is a machine learning paradigm that can rapidly examine new tasks using prior knowledge and learn using supervised knowledge from a limited number of samples (Wang et al., 2020). For this reason, Few-Shot learning is referred to as a Meta Learning challenge. Meta Learning is recommended as a framework for dealing with the difficult Few-Shot learning environment. The fundamental concept is to use a large number of comparable multi-step tasks to figure out how to adapt the core learner to a new activity for which there are only a few labelled samples (Sun et al., 2019). Although being able to classify with very little data is a massive

challenge, the few-shot learning approach is illustrated to provide classification with a high degree of accuracy (Fei-Fei et al., 2006) (Fink, 2004).On the other hand, since there is relatively little data in each class, contrastive learning can be a promising approach to deal with this problem, allowing it to train the model to learn a lot about the data without any explanations or labels. Rather than learning a signal from individual data samples, contrastive learning maximizes agreement on multiple augmented views of the same image, while learning by negotiating instances generated by different base images (Ni et al., 2021) (Le-Khac et al., 2020).

In this study, we apply our baseline model for the ImageNet dataset, for which we have limited data, and perform our experiments with different data augmentation methods and dropout. In addition, we apply the contrastive learning method, which learns by comparing the positive pairs of similar inputs and the negative pairs of dissimilar inputs mentioned above, and Prototype networks (Snell et al., 2017) and Reptile algorithm (Nichol et al., 2018) from Meta Learning categories to the classification process in our dataset, which has a limited number of labelled training data.

The sections of the study that follow are arranged in the following order: Section 2 describes the properties of ImageNet dataset as well as the tasks to be completed. We explain how these solutions tackle the problem in section 3 by outlining the approaches we propose for limited data. The use of these approaches and the specifics of their application are described in the fourth chapter, along with the results and their interpretations. The literature on research in this area is reviewed in section 5, and studies that contain solutions to the problem are examined. Finally, in section 6, we conclude the study and offer recommendations for future research.

2. Data set and task

In this study, we use a subset of Imagenet data as the dataset. Unlike the original Imagenet, which has over 1 million training images, 50,000 validation images, and 100,000 test images, our dateset has 1000 classes with 50 image training samples each. As a result of this condition, we are confronted with the limited data problem, which is one of the most serious issues. A model with too many classes and not enough samples in each class is difficult to train and causes a generalisation issue, resulting in low accuracy. Figure 1 illustrates several samples from the dataset.

Metrics to measure accuracy The goal of this research is to explore different types of meta learning techniques to better understand how it helps solve the limited data problem by having good generalization ability. Since we are using meta learning protocol we use different way of estimating the test accuracy. In the normal learning framework we use train data to train the model and test data to evaluate its performance. But in case of meta learning since we divide the task into smaller classification problems and use the parameters of one problem to solve the classification problem on unrelated data. The smaller datasets are referred to as support set and query set.

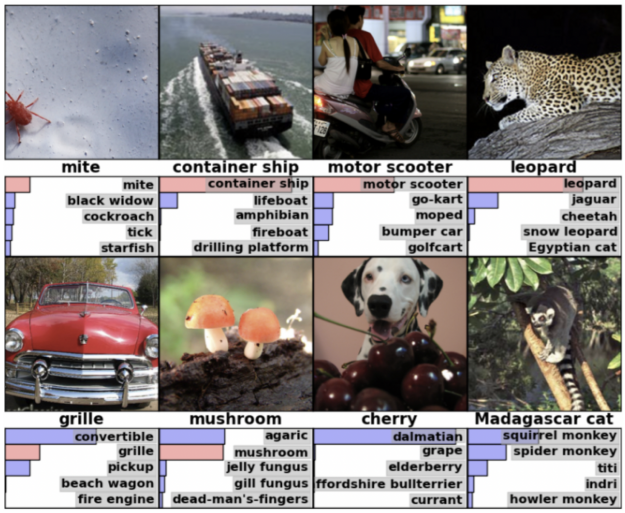

Figure 1. Sample images from the dataset

3. Methodology

We tried adopting different methods and loss functions in our approach to pre-train the networks before supervised training on the full data.

3.1. Contrastive learning of unsupervised representations

Contrastive learning is a form of deep metric learning and was introduced in (Hadsell et al., 2006). The loss function for this approach is formulated in such a way as to pull the representations of different augmented versions of the same image closer while pushing apart the representations of different images apart. There is also a supervised extension for this loss (Khosla et al., 2020) that pushes apart the representations of the images of different classes, while reducing the distance between the features of the images of the same class or category. Its application for pre-training the representations before applying supervised learning through cross-entropy loss has even beaten the performance of network trained with cross entropy loss on labels alone (Chen et al., 2020). We adopt the following loss functions to learn a dense representations of images in an unsupervised regime before few-shot learning.

$$l(i,j) = -log[\frac{e^{s(i,j)}}{\sum_{(1,k)} e^{s(i,k)}}] \quad (1)$$

where s(i,j) is the cosine similarity between the normalized outputs of the encoder for input images i and j.

$$s(i,j) = \frac{z_i * z_j}{||z_i|| * ||z_j||} \quad (2)$$

, where ||.|| denotes the $l2$ norm of the encoder outputs. Here, the images i and j are two augmented versions of the

same image, while k is the set of all the images passed in the batch. Thus, if t and $t+1$ images in a batch of size N are augmented versions of the same image, $i and j$ in 1 are then $1, 2, 3, 4, ...N-1, N$ while the $k's$ in denominator are the indices for remaining images. This loss bears a resemblance to the cross entropy loss and will be higher if the augmented versions of the same image will be far apart in representation space, thus removing For training the encoder, we first create two augmentations of each image and pass both the versions of each image in the batch. The numerator term $s(i, j)$ is then the encoder output of the augmented versions of an image while the denominator contains the encoder outputs for all other images. Thus, this loss will remove the variance in the representations due to these augmentations and make the representations more invariant to rotation, flipping and other simple transformations which were applied as part of augmentation while maintaining inter-class separation. The representations were first extracted through a Resnet50 encoder over which global average pooling was applied to extract aggregated statistics from the image. These features were passed though a projection head which consisted of two densely connected relu layers that projected the features in a $R \in 2048$ dimensional space. The loss function described above was then applied over these features extracted from a batch of images where features of the augmented versions of the same image were pulled closer by using them in the numerator of the loss functions while pushing them apart with dense representations of other images which were in the denominator. After training the network, the projection head was removed, and the underlying features can be directly used for supervised learning.

3.2. Meta Learning

Meta learning refers to the process of learning what to learn from the data. In the recent years to solve the problem faced by limited data meta learning algorithms played a key role. In this report we discuss two types of meta learning techniques gradient based meta learning and loss based meta learning. The gradient based meta learning techniques use gradient descent to learn common features from different tasks. Some of the example algorithms include MAML (Finn et al., 2017), Reptile (Nichol et al., 2018) etc. Whereas loss based meta learning algorithms use features produced by the network as representations and try to reduce loss based on the fact that the same label data should be nearer to each other in feature space. Some of the example algorithms are Matching networks(Vinyals et al., 2016), Prototypical networks(Snell et al., 2017) etc. In this report we further concentrate on exploring algorithm from each category of meta learning techniques (Prototypical networks, Reptile) and try to understand its performance on the given dataset in few shot learning regime.

3.2.1. Prototypical networks

Prototypical networks (Snell et al., 2017) is a metric space based solution provided for few-shot classification problem where the classifier is expected to generalise to new classes given only a small number of examples of each of these new class. A prototype representation for each class is learned in the metric space and when a new input image is shown to the network, an embedding of this image is mapped into this metric space (known as "query point") and using Euclidean distance, classification is done by finding the class prototype nearest to this query point.

In few-shot learning, for a given limited dataset, the number of classes are divided as 'train classes' and 'test classes' such that the episodic training is done on the train classes. Then, the test classes (the classes that the model has never seen before) is used to measure how much the model can generalise to unseen data. Prototypical Networks are known to reflect simpler set of assumptions which the model uses when dealing with unseen data, aka Inductive Bias. Keeping the overlapping between the two types of classes (train and test) is a good strategy to follow in such learning methods.

The prototype representation for a particular class is computed by first using Deep Neural Networks to learn a non-linear mapping which can be used to get a embedding (feature vector). Mean of such embedding spaces of the same class support set is the class's prototype (refer equation 3). Classification of the query set is then done by finding the nearest class prototype.

$$c_k = \frac{1}{S_k} \sum_{(x_i, y_i) \epsilon S_k} f_\phi(\mathbf{x}_i) \tag{3}$$

c_k is an M-dimensional representation or class's prototype
ϕ is the learnable parameters of an embedding function f_ϕ

The loss, J, is updated using the following formula, where d is the distance calculated between a query data point and prototype of each class.

$$J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_\phi(\mathbf{x}), c_k) + log \sum_{k'} exp(-d(f_\phi(\mathbf{x}), c_k)) \right] \tag{4}$$

The softmax equation given below is used to select the class with the maximum negative distance i.e. minimum distance. This helps classify the query point to the class whose prototype is at minimum distance.

$$p_\phi(y = k|\mathbf{x}) = \frac{exp(-d(f_\phi(\mathbf{x}), c_k))}{\sum_{k'} exp(-d(f_\phi(\mathbf{x}), c_{k'}))} \tag{5}$$

3.2.2. Reptile

Reptile algorithm (Nichol et al., 2018) is a gradient based technique where we try to learn the best initialization of parameters of a network so that the model generalizes easily for new tasks. In this method we optimize the following loss

$$min_\phi = E[L_\tau(U_\tau^k(\phi)] \tag{6}$$

where the expectation is over the loss function and ϕ are the model parameters. Thus, U_τ^k performs a gradient step over

the original parameters for the weights of each task(which is the class in our case) τ and brings the weights for that task into an initialization such that it generalizes well for that particular class. This can be written formally as

$$U_\tau^k(\phi) = \phi - \frac{\partial L_\tau}{\partial \phi} \tag{7}$$

Each task will then have its own copy $\hat{\phi}$ obtained through U_τ^k of the original parameters ϕ and the optimization is done for these parameters in the meta training stage. The U_τ^k in this case that generates the weights for task τ is k steps of gradient descent for the loss of the task τ. After the copies for each task has been created, the original parameters ϕ are updated in the direction of each copy as

$$\phi = \phi + \epsilon \frac{1}{n} \sum_i [\hat{\phi} - \phi] \tag{8}$$

, where i is the ith task. The algorithm is as follows:

Algorithm 1 Reptile (serial version)

Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for

Figure 2. Reptile algorithm (Nichol et al., 2018)

4. Experiments

4.1. Baseline

For this mini-project we have done experiments using both tensorflow and pytorch frameworks so we recreated the baseline model with same training protocol as pytorch version in tensorflow as well. To understand how the image size makes difference we have experimented with different image sizes such as (256,256) ,(224,224) ,(128,128).

4.1.1. Training Protocol for baseline

- Data was loaded using dataloaders and pre-processed by normalizing every images with mean and variance values according to imagenet dataset(Russakovsky et al., 2015).
- The training data was divided into train and validation set in ratio of (80:20).
- The different augmentation techniques used include horizontal flip(mirroring of images) and random crop the image to size (224,224) while training and center crop was applied during validation and testing.
- Only horizontal flip(mirroring the images) augmentation was done on random images during training.
- Sparse categorical cross entropy was used as loss function along with a combination of SGD, weight decay was applied as optimizer to the base resnet-50 architecture.
- As for the learning rate we used decreasing learning rate which changes every 30 epochs by a factor of 0.1 (i.e) starting learning rate = 0.1 and end learning rate was 0.001 at epoch 90.
- The model was then trained for 90 epochs and the best model was chosen based on the validation accuracy(This was achieved at epoch 64).

Results/Interpretation: We used the best model chosen based on validation accuracy for all the experiments and then evaluated on test set. It was observed that when image size of (256,256) was used the test accuracy was 26.14% and with size(224,224) the test accuracy was reduced to 23.5%. When the image size was further decreased to (128,128) the test accuracy dropped to 10.73%. This is to be expected because as we decrease image size the performance decreases as we are removing high frequency information from the image. Due to this there is high chance that no of false negatives decrease and no of false positives increase which in-turn affects the performance of the model.

4.2. Analysis of data

: To further understand where the baseline model fails we try to analyze it in detail in this section. It is observed that our baseline model is confident in some cases and not in others(3).

- We find that there are around 53 classes where the original baseline fails completely (i.e) gives 0% test accuracy.
- Where as the model performs best for these 5 classes (986, 95, 340, 946, 376) i.e test accuracy > 40%.
- There are around 39 classes which have test accuracy between 30% and 40%.
- It is observed that the highest no of classes have >10% test accuracy(455 classes).
- When we observed the above classes we find that these classes had very poor training images (i.e) there were more than 2-3 objects in the image and the label was given to the object which was very small. It is understood that this is what we get from real life data images.
- And since we are only using the top 1% accuracy measure in this report we need to understand that we do not get drastic results from these experiments.
- A more suitable problem would be detection of objects rather than classification for the given dataset.

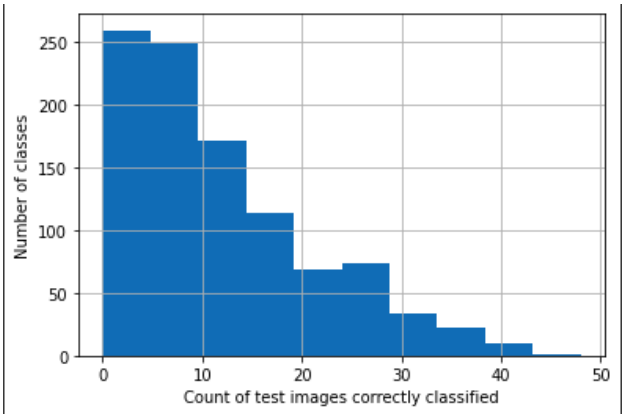

Figure 3. Graph showing no of correctly classified images by the baseline

4.3. Experiment with dropout

To investigate the importance of the features learned by model we tried experimenting with dropout. For this experiment we used the baseline which takes input size(256,256). The dropout layer was added before the fully connected layer in the resnet50 architecture.

4.3.1. Training protocol

In this experiment the baseline model was frozen upto global average pooling layer and then a dropout layer with probability of 0.2 was added along with a fully connected layer(1000 neurons) with softmax activation. It is trained for 20 epochs with same training protocol as baseline model and the best model was chosen based on val accuracy. Then we fine tuned this new model by unfreezing all the weights of baseline and using a very low learning rate (1e-5) with adam optimizer for 10 epochs.
Results/Interpretation It was observed that the new model with dropout gave a test accuracy of 20.26% whereas the baseline performance was 26.14%. It is to be observed that even when dropout of 20% was used the performance of the model decreased by 6% proving that the features it learned is important.

4.4. Experiments with contrastive losses

We applied contrastive losses on the top of baseline networks to learn features that are more invariant within a classes while maintaining inter class separation of these features. For each image, we created two augmentations of it using the following augmentations: random flipping, random crop of patch size (224,224) of the original image, rotation between (-0.5,0.5) radians and color jittering and random contrast. We experimented with both unsupervised and supervised contrastive losses. After learning the features from the contrastive losses, we removed the last layer of the encoder which were fully connected dense layers of dimensions $R \in 2048X2048$ and passed the features through a softmax layer that was used to optimize the cross-entropy loss on the whole dataset. The results for both the losses are shown in. Since, these losses use Noise contrastive estimation (NCE) which requires large number of negative samples to be passed in each batch for correct estimation and optimization of target probabilities and we were limited to a batch size of 256, the features trained from these methods could not achieve competitive performance compared to the baseline and few shot methods. We make the following conclusions from this experiment.

- Contrastive learning only works well with larger batch sizes of upto 4096 as used in (Chen et al., 2020) for proper estimation of noise contrastive estimation and training for longer durations. Since, we were limited to batch size of 256, we believe that this estimation was poor and the representations of the images could not be correctly estimated. The final accuracy that we could get on test set was only 2

- Adding more augmentations did not help much and keeping the rotation range and random crops ratio to be small gave better results, as the contrastive loss decreased more steadily when the magnitude of augmentations were small, whereas in the previous case it varied a lot and did not decrease much. This suggests that picking good augmentations that help the model generalize is really crucial as augmentations that are too different from the actual distribution of test images can hurt model performance.

4.5. Experiment with Reptile algorithm

As mentioned in the methodology section we try to investigate the use of reptile algorithm in few shot scenario. It is to be noted that we follow the meta learning training framework for this experiment to evaluate the performance of the model. The aim of this experiment is to know whether reptile algorithm can give a good initialization values for the network which can be further fine tuned on other tasks in each meta iteration. (Note meta iterations signify number of times we are sampling from the original dataset i.e each meta iteration defines new classification problem we are trying to generalize to)

4.5.1. Training protocol

- We load the dataset according to the problem defined in the methodology section. For our experiment we used 5 shot on 5 classes so we sample 25 images each time for training from training set and 1 image for testing the model from test set.

- We do not use any pre-trained model for this few shot learning technique because it violates the premise of few shot problem. Therefore we use plain resnet50 (He et al., 2016) with weights = None and classes = n (n= 5).

- There are no transformations applied on dataset and there is no scope of overfitting because the weights

get updated on each meta iteration which is a new classification task with a new mini-dataset(25 images).

- We use SGD with learning rate of 0.003 as mentioned by the authors in the paper(Nichol et al., 2018) and based on our restrictions of processing we use inner batch size 10 ideally it should be above 40 (Nichol et al., 2018).

- We then follow the mentioned algorithm in the methodology section.

- We do this for 2000 meta iterations and get an average test accuracy at the end.

Results/interpretations: We can look at the mini-dataset (6) to see what classes the model is training on at some random meta iteration. It is observed that the train accuracy averages around 70%(note this is on 5 class classification problem) and test accuracy around 40%. Some of the example images are below where the test labels are produced. But these results do not provide enough insights to the context of our problem. To know the performance of reptile model we use it as an initialization technique as proposed by the authors((Nichol et al., 2018) in the next section.
It is to be noted that we do not use reptile as a feature extractor method because there is no update taking place in parameters for each inner loop step in the algorithm. This is also mentioned by authors of (Goldblum et al., 2020) where they find that the reptile algorithm when used as feature extractor performs worst than the classical trained model using same architecture.

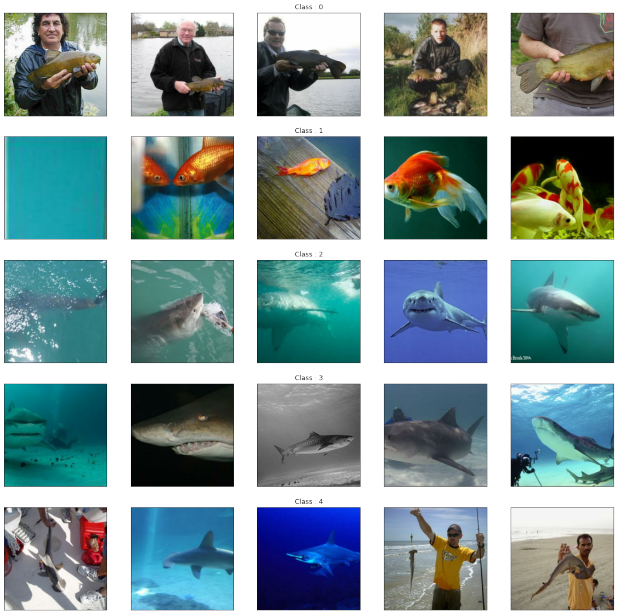

Figure 4. Mini dataset produced for each meta iteration in reptile algorithm

Figure 5. Example mini test set with labels and predictions

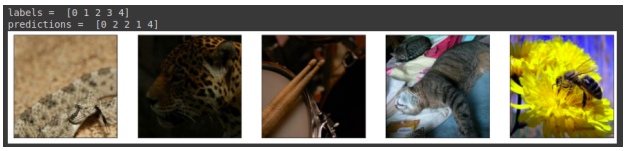

Figure 6. Example mini test set 2 with labels and predictions

4.6. Baseline pretrained with reptile algorithm

It was clear from the above experiment that the reptile was giving decent results even when the number of parameters did not match the original paper. To further understand its performance on whole test set we now use this resnet50(trained using reptile) and run with the baseline protocol to observe results.

4.6.1. Training Protocol

- The model trained on reptile has 5 output neurons because we were doing 5 way 5 shot learning in the above experiment. To run it on whole test set we need to change the final layer we add to the network.

- Now a dense layer of size 1000 is added to the network by removing the output layer(containing 5 neurons).

- Since we now have a network architecture we desire we train it using the baseline protocol adding augmentations as well to the images.

Results/Interpretation It is observed that the training time is decreased by a huge amount (by around 10 hrs). The new model reaches a highest validation accuracy of 17% and it gives 17.08% accuracy on test set. It is also observed that there is a reduction in generalization gap(train_loss - val_loss) in the original baseline the gap was equal to 1.919 where as the new model has generalization gap of 0.523. This also proves that the baseline model when pretrained with reptile model was able to generalise well. It was still not able to beat the baseline performance and the reason is because the reptile model was not fully trained. According to the authors (Nichol et al., 2018) we need to run for 100,000 meta iterations where as we ran it for only 2000 meta iterations because of memory constraints in colab pro. By this experiment we can conclude that the weight initialization of the model also has a huge affect on the performance of the model so with a good starting weights the model will be able to generalize well and also have an improved performance. "

4.7. Prototypical networks

- Instead of using the 4 block CNN architecture mentioned in the original paper (Snell et al., 2017), we use a ResNet50 (He et al., 2016) backbone. In order to get a non-linearly mapped embedding represention for each image, we remove the final softmax layer from the original ResNet50 model and use the average pooling layer right before it to obtain the embeddings. A non pre-trained ResNet50 model has around 23M parameters. Due to resource constraints and also to avoid overfitting our model by training limited data on such a high trainable parameter network, we freeze all the layers except the final convolution and bias weights. Hence, the total number of trainable parameters in our model is 1052672 and the final embedding vector size is 2048.

- For meta learning tasks it is important to segregate the train and test classes such that the overlapping between them is kept to a minimum. The goal is to train a network using a certain set of training classes and test the performance of this network using classes that the model has never seen before. This is to encourage the "learning to learn" concept of meta-learning (Hospedales et al., 2020). Thus, we first segregate the 1000 class available into 700 train classes, 200 validation classes and 100 test classes. On observing the data, we noticed that consecutive classes are related i.e. if we were to segregate the classes sequentially there would be a bias learned by the model based on the pattern observed between 2 consecutive classes. To avoid this, we randomly pick classes for train/val/test.

- While training the model, we experiment with different image sizes (224*224 and 128*128). For training, we randomly extract a sample 20-way ie. 20 unique classes with 5 random images in both support and query set of each class, and do 2000 episodic training in each epoch. We train the experiments for total 5 epochs. We use validation class set at the end of each epoch to check the performance improvement after every epoch. Once we decide on the best weights looking at the validation class set performance, we test the model on the 100 test classes and then do the "final testing" using the provided 50000 images test dataset.

- The extracted sample is forward propagated through the network and the support set of each class is used to compute that particular class's prototype. This is done by taking the mean of the embedding space of this support set. After getting every class's prototype, loss is computed using the query set. This loss is computed by first calculating euclidean distance (as mentioned in (Snell et al., 2017)) between the query points and prototype and then applying log softmax to the negative distance vectors and predicting the class with the highest probability. The training/val accuracy and loss plot for image sizes 224*224 and 128*128 can be observed in figure 6 (Accuracy plot) and 7 (Loss plot).

- During final testing, we do Episodic testing, where we extract a sample by sequentially going through each class (from 0 to 999). We use less number of ways as compared to training. The authors of (Snell et al., 2017) clearly mention that better performance for prototypical network was observed when the number of ways for testing is lower that than for training but the number of images in support and query set should remain the same for training and testing. We experimented with 10, 5, 4 ways and as expected, the test accuracy dropped as the number of ways increased (refer figure (8)). After extracting the sample, we pass it through the model the same way we did for training except this time we do not back propagate to update the weights of the model.

  Results/Interpretation While experimenting with different data sizes we conclude that image size matters in getting good accuracy(training/testing). Even though the computation time required by image size 224 was noted to be 50% more than that for image size 128, as seen in figure 6, the training accuracy is consistently 10% more for each epoch for image size 224. The feature vector (2048 dimensional) representing the image of size 224 seems to represent the image much better than that of size 128. The validation accuracy pretty much remains the same after every epoch but image size 224 has higher accuracy there too. We observe We also observed the authors note about using less number of ways while testing to be true. As seen in fig 8, the test accuracy drops as n_way size increases. Apart from the test accuracy measured using the test set (of 50k images), as mentioned in the previous section, 100 classes were kept aside for testing. These 100 classes were specifically used to measure the meta-learning capability of model i.e. the model's ability to learn unseen classes. This accuracy was noted to be 70%, 63.6% and 56.80% for 4-way, 5-way and 10-way respectively.

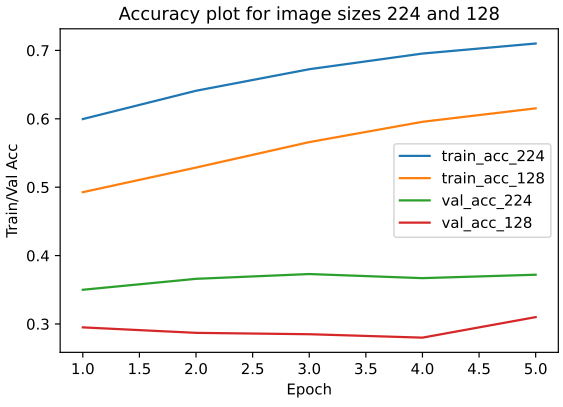

Figure 7. "Accuracy plot of train and val set for image sizes 224 and 128."

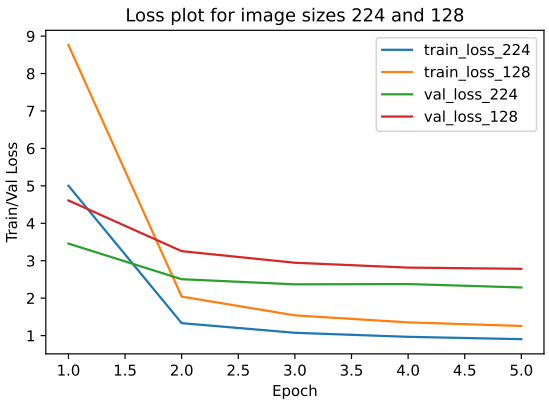

Figure 8. "Loss plot of train and val set for image sizes 224 and 128."

n-way	test accuracy
4	0.2526
5	0.2114
10	0.1036

Figure 9. "Variation of test accuracy score wrt n_way for image size 224"

5. Related work

Many methods are proposed in the training of CNN models to overcome the limited data problem while also preventing the overfitting problem. Numerous research in the literature utilise various regularisation strategies to try to solve these issues based on the datasets they use (Krizhevsky et al., 2012)(Mikołajczyk & Grochowski, 2018)(Shorten & Khoshgoftaar, 2019). To deal with these issues, data augmentation is one of the most often utilised editing formats in deep CNN training. The limited data problem is addressed with this approach using augmentation methods like as flipping, rotating, clipping, and adding noise to the existing data set, as well as overfitting the overfitting problem. Deep CNN training often employs random flipping and random cropping approaches (Zhong et al., 2020). In addition, Krizhevsky et al.(Krizhevsky et al., 2012) utilising the Imagenet dataset prevented overfitting by using image translations and horizontal reflections techniques in the data augmentation part to artificially increase datasets. In addition to applying traditional image transformations such as rotating, cropping, zooming, and histogram-based approaches, researchers provided a data augmentation method based on the image Style Transfer method in (Mikołajczyk & Grochowski, 2018). The content of one base image is combined with the views of the others in this approach to create new images with high perceptual quality, and the newly formed images may be used to pre-train the supplied neural network to improve the efficiency of training process. Shorton and Khoshgoftaar (Shorten & Khoshgoftaar, 2019) discuss the necessity of improving the size and quality of training datasets to construct stronger Deep Learning models in their evaluation of research recommending Data Augmentation approaches as a solution to the limited data challenge. They emphasised the importance of how to expand limited datasets to take advantage of the features of big data while providing information about methods and properties such as geometric transformations, colour space augmentation, mixing images, feature space augmentation, kernel filters, adversarial training, random erasing, generative adversarial networks, meta-learning and neural style transfer. At the same time, they mention that the overfitting problem that might arise owing to a lack of data can be addressed by using data augmentation techniques to variety the dataset.

Studies on unsupervised representation learning from images, on the other hand, have shown promising results in recent years by including the notion of contrastive learning. SimCLR, for example, proposed by Chen et al.(Chen et al., 2020), learns features by comparing images following a data augmentation composition. Negative pairs feature two separate images, whereas positive pairs are formed by sampling two images for the same image after using various augmentation algorithms. SimCLR also narrows the performance difference between unsupervised and supervised pre-training representations in linear classifiers. In the ImageNet dataset, this strategy looks to outperform existing self-supervised learning methods substantially. Furthermore, in (Wang & Qi, 2021), the researchers proposed Contrastive Learning with Stronger Augmentations (CLSA) as a method to supplement existing comparative learning approaches, and experiments on the ImageNet dataset and downstream datasets show that information from strongly augmented images can significantly improve performance.

The researchers proposed prototypical networks for a few-shot classification problem, which has only a small number of samples from each new class, and where a classifier should generalise to new classes that are not seen in the training set, in (Snell et al., 2017), which is one of the studies with limited data problems such as miniImageNet. This method focuses on learning embeddings that alter data in a way that a fixed closest neighbour or linear classifier can recognise it. By comparing the Prototypical Network to other networks, researchers working with the miniImageNet dataset, which comprises 100 classes and 600 samples in each, have proven that the Prototypical Network has produced successful outcomes. Similarly, using the Improved Prototypical Networks (IPN) technique with the ResNet-10 backbone, the researchers (Ji et al., 2020) achieved classification accuracy of 56.18% for 1-shot and 74.60% for 5-shot using the dataset with the same features. Furthermore, the Reptile technique, another Meta Learnig algorithm, has been suggested to provide effective results in a variety of datasets (Kedia & Chinthakindi, 2021), including the study of Nichol et al. (Nichol et al., 2018) with the miniImageNet dataset.

6. Conclusions

To conclude we explored various techniques to solve the problem of limited data. We learned that image size plays a huge role in classification accuracy for the given dataset. We analyzed different type of pre-training strategies and conclude that they always do not lead to performance, the types and range of augmentations and the batch size play an important role for successful pre-training. We also found out that meta learning techniques perform better when trained for good amount of time. In particular we find that the reptile algorithm if trained for good amount of time might bring the weights to a good initialization which in turn helps improve model generalization and performance. For prototypical network, we conclude that the model's meta learning capability is evidently good, especially when the number of ways used for testing is less than that used while training. We also conclude that image size matters a lot for better feature vector (2048 dimensional) representation. This was evident from the consistently higher accuracy obtained from image size of 224 than that for size 128.

References

Chen, Ting, Kornblith, Simon, Norouzi, Mohammad, and Hinton, Geoffrey. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Fawzi, Alhussein, Samulowitz, Horst, Turaga, Deepak, and Frossard, Pascal. Adaptive data augmentation for image classification. In *2016 IEEE international conference on image processing (ICIP)*, pp. 3688–3692. Ieee, 2016.

Fei-Fei, Li, Fergus, Rob, and Perona, Pietro. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Fink, Michael. Object classification from a single example utilizing class relevance metrics. *Advances in neural information processing systems*, 17, 2004.

Finn, Chelsea, Abbeel, Pieter, and Levine, Sergey. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Fonseka, Deshan and Chrysoulas, Christos. Data augmentation to improve the performance of a convolutional neural network on image classification. In *2020 International Conference on Decision Aid Sciences and Application (DASA)*, pp. 515–518. IEEE, 2020.

Goldblum, Micah, Reich, Steven, Fowl, Liam, Ni, Renkun, Cherepanova, Valeriia, and Goldstein, Tom. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning*, pp. 3607–3616. PMLR, 2020.

Hadsell, Raia, Chopra, Sumit, and LeCun, Yann. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Halevy, Alon, Norvig, Peter, and Pereira, Fernando. The unreasonable effectiveness of data. *IEEE intelligent systems*, 24(2):8–12, 2009.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hospedales, Timothy, Antoniou, Antreas, Micaelli, Paul, and Storkey, Amos. Meta-learning in neural networks: A survey. *arXiv e-prints*, pp. arXiv–2004, 2020.

Ji, Zhong, Chai, Xingliang, Yu, Yunlong, Pang, Yanwei, and Zhang, Zhongfei. Improved prototypical networks for few-shot learning. *Pattern Recognition Letters*, 140: 81–87, 2020.

Kedia, Akhil and Chinthakindi, Sai Chetan. Keep learning: Self-supervised meta-learning for learning from inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 63–77, 2021.

Khosla, Prannay, Teterwak, Piotr, Wang, Chen, Sarna, Aaron, Tian, Yonglong, Isola, Phillip, Maschinot, Aaron, Liu, Ce, and Krishnan, Dilip. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Le-Khac, Phuc H, Healy, Graham, and Smeaton, Alan F. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

Look, Andreas and Riedelbauch, Stefan. Dealing with limited access to data: Comparison of deep learning approaches. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Mikołajczyk, Agnieszka and Grochowski, Michał. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pp. 117–122. IEEE, 2018.

Ni, Renkun, Shu, Manli, Souri, Hossein, Goldblum, Micah, and Goldstein, Tom. Contrastive learning is just meta-learning. In *International Conference on Learning Representations*, 2021.

Nichol, Alex, Achiam, Joshua, and Schulman, John. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Oquab, Maxime, Bottou, Leon, Laptev, Ivan, and Sivic, Josef. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.

Pang, Yanwei, Sun, Manli, Jiang, Xiaoheng, and Li, Xuelong. Convolution in convolution for network in network. *IEEE transactions on neural networks and learning systems*, 29(5):1587–1597, 2017.

Qiao, Shijie and Ma, Jie. A face recognition system based on convolution neural network. In *2018 Chinese Automation Congress (CAC)*, pp. 1923–1927. IEEE, 2018.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Shorten, Connor and Khoshgoftaar, Taghi M. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

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

Snell, Jake, Swersky, Kevin, and Zemel, Richard. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Sun, Chen, Shrivastava, Abhinav, Singh, Saurabh, and Gupta, Abhinav. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Sun, Qianru, Liu, Yaoyao, Chua, Tat-Seng, and Schiele, Bernt. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 403–412, 2019.

Vinyals, Oriol, Blundell, Charles, Lillicrap, Timothy, Wierstra, Daan, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Wang, Xiao and Qi, Guo-Jun. Contrastive learning with stronger augmentations. *arXiv preprint arXiv:2104.07713*, 2021.

Wang, Yaqing, Yao, Quanming, Kwok, James T, and Ni, Lionel M. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53 (3):1–34, 2020.

Xu, Yi, Noy, Asaf, Lin, Ming, Qian, Qi, Li, Hao, and Jin, Rong. Wemix: How to better utilize data augmentation. *arXiv preprint arXiv:2010.01267*, 2020.

Zhong, Zhun, Zheng, Liang, Kang, Guoliang, Li, Shaozi, and Yang, Yi. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 13001–13008, 2020.