Attempting Low Resource Image Classification

Ashwin Bhat, Eric Chiang, Dylan Lewis

***Abstract*— When working with neural networks, as the amount of available data increases, the performance also increases (to a certain point). However, when it is impossible to obtain abundant amounts of training data, the neural network's ability suffers. Without enough data, it is unable to learn useful features and because of this, its performance is poor. In this paper, we explore the effects of data augmentation on a specialized "Where's Waldo" data set and how it affects the performance of transfer learning neural models.**

I. INTRODUCTION

With the recent success of neural networks, in particular deep networks, the problem of image classification has had growing success. Many different model architectures are producing state of the art results while have very different model architectures. However, none of these models can circumvent a pivotal problem to training a neural network: data. Without enough training data, the network is unable to learn useful features of the data, and thus its accuracy plummets. There are many ways to try and circumvent this problem of not having enough data, for example going out and finding more or augmenting your original data. In particular, we are interested in the second solution.

For our experiments we focus on the well known children's puzzle book "Where's Waldo?" The goal of the book is to find in very busy images a man named Waldo, who has a red and white striped beanie and round glasses. Aside from the image being very cluttered with other characters, objects, and details, Waldo's body is often partially or fully occluded. What we hope to achieve is to train a model who given an image can tell whether or not Waldo is present in the image (finding him is left as an exercise to the reader). This serves as a very interesting dataset for the problem of image classification as Waldo does not show up the same way in any two photos, and is more than likely obfuscated in some way. This means that to correctly classify an image, the model must learn a rich feature representation of Waldo.

II. DATA PREPERATION

The original Waldo Dataset contains images of sizes 256x256, 128x128, and 64x64 in full color, black and white, and gray scale. For our experiments we chose to use the full color 128x128 dataset. The dataset is split into two parts: images with Waldo, and images without. For exact counts of the number of images in each dataset see Table 1. Originally, not only is the dataset is very skewed towards pictures without Waldo, but we must also split this small set images up into train, test, and validation sets. With only 27 original images containing Waldo, a decision had to be made about how to split the data. The solution we decided on was to use a test set that was a majority duplicate positives. In addition we explored two different experiments, one where we restricted ourself to training only on the original dataset, and one where we allow ourself to augment the original data. The way we created our splits are as follows:

- Choose two different subset of images containing Waldo. For our purposes they were of size 4. These two subset create a basis for our validation and test sets.
- Choose an equal amount of images not containing Waldo and add them to the validation and test set. Now we have completed test, train, and validation sets for our first experiment.
- Create the augmented data set by augmenting images only in original data set.
- Use the augmented test data for both experiments using only the original dataset, and experiments with augmented data.

The reason we chose to use a synthetic test dataset was because the lack of examples of Waldo in the original. Our validation dataset of 4 Waldo images and 4 not Waldo images is very small, and to make sure that we got accurate statistics on the test data, it was out of the question to use a test dataset of the same size. In addition, both the augmented experiment and baseline experiment must use the same test data to keep results fair. To try and alleviate these problems we decided to use augmented data for the test set, but still make sure the pictures were never seen during training or test in both experiment's cases.

TABLE I
TABLE 1. SIZE OF DATA SETS

	Baseline		
	Waldo	Non-Waldo	Total
Train	19	814	833
Validation	4	250	254
Test	277	253	554

	Augmented		
	Waldo	Non-Waldo	Total
Train	763	814	1577
Validation	278	250	528
Test	277	253	530

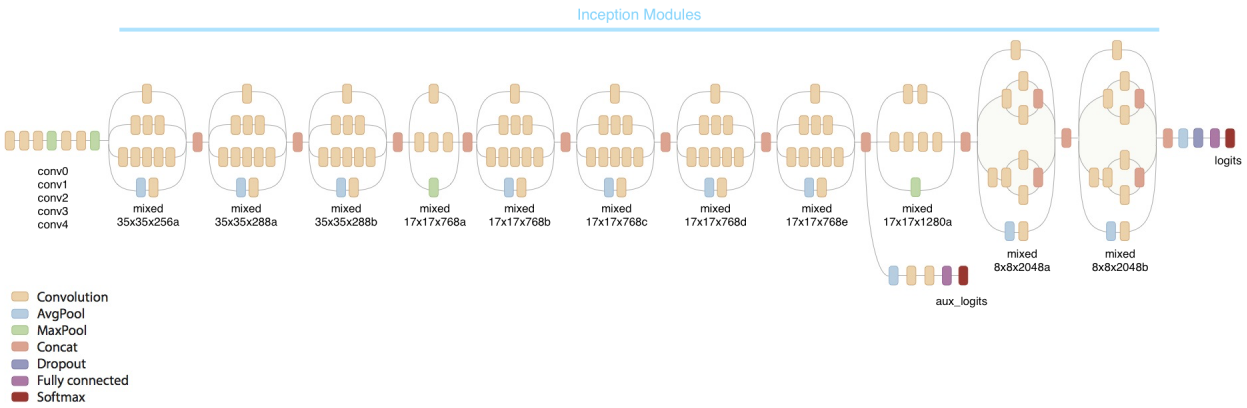


Fig. 1. Inception Model Architecture

III. DATA AUGMENTATION

It has been show that augmenting datasets can be helpful for the use of image classification [1]. In particular, for our problem we are dealing with the problem of inbalanced training data. To combat this problem, we attempt to use various oversampling techniques. To oversample, we take images that are under represented (in our case, pictures with Waldo) and find ways to augment them. Below we out line how we oversampled our dataset.

IV. DATA AUGMENTATION: FIRST PASS

To augment the data, for each image we took a single column of pixels and moved them to the opposite side. We repeated this transformation for every column in the image, thus giving a "roll" effect over the whole set of images. In addition we also did this for each row. A final vertical reflection was also applied to each image. The reason we decided on this type of augmentation is that it allowed for a robust, but not complicated augmentation that would allow us to exponentially grow the number of training examples. In addition, due to the nature of rolling the image, the location of Waldo will change between points where he is completely contained in the image, and points where part of him is on a border. We believed that this would help in learning a more context independent representation of Waldo. Theoretically we should have been able to create 2.4 million unique images of Waldo and 13 million non-Waldo images. However, due to space and resource limitations, we were unable to carry out experiments with this dataset, and instead had to rely on a secondary augmentation technique.

V. DATA AUGMENTATION: SECOND PASS

The second pass at data augmentation allowed us to create a dataset that was able to be trained with our available resources. To create the dataset, we first created copies of the Waldo images, then applied a multitude of different transforms to them. We used: reflections, rotations, translations, and shearing to create the dataset that would become our final augmented data. It is important to note that these augmentations were only used to boost the number of Waldo images so that it could match the number of non-Waldo images.

VI. MODELS

For the actual models we trained, we chose to use transfer learning techniques using Inception v3 [2], ResNet [3], and DenseNet [4] model architectures. The reason we chose to use Transfer Learning is due to its ability to not need abundant amounts of data to train on, which we believed would increase performance on the finding Waldo task. Below we give a very brief summary of the different model architectures used.

A. *Transfer Learning*

The goal of transfer learning is to take a preexisting trained model and retrain it to complete a more customized task. In our case we are using transfer learning to take models that were trained to classify images on ImageNet and retrain it to classify our Waldo images. The way we carried out transfer learning is by continuing training the weights of the model pretrained on ImageNet. That is, we don't start with a random initialization, but rather use the pretrained model as a basis to train ours.

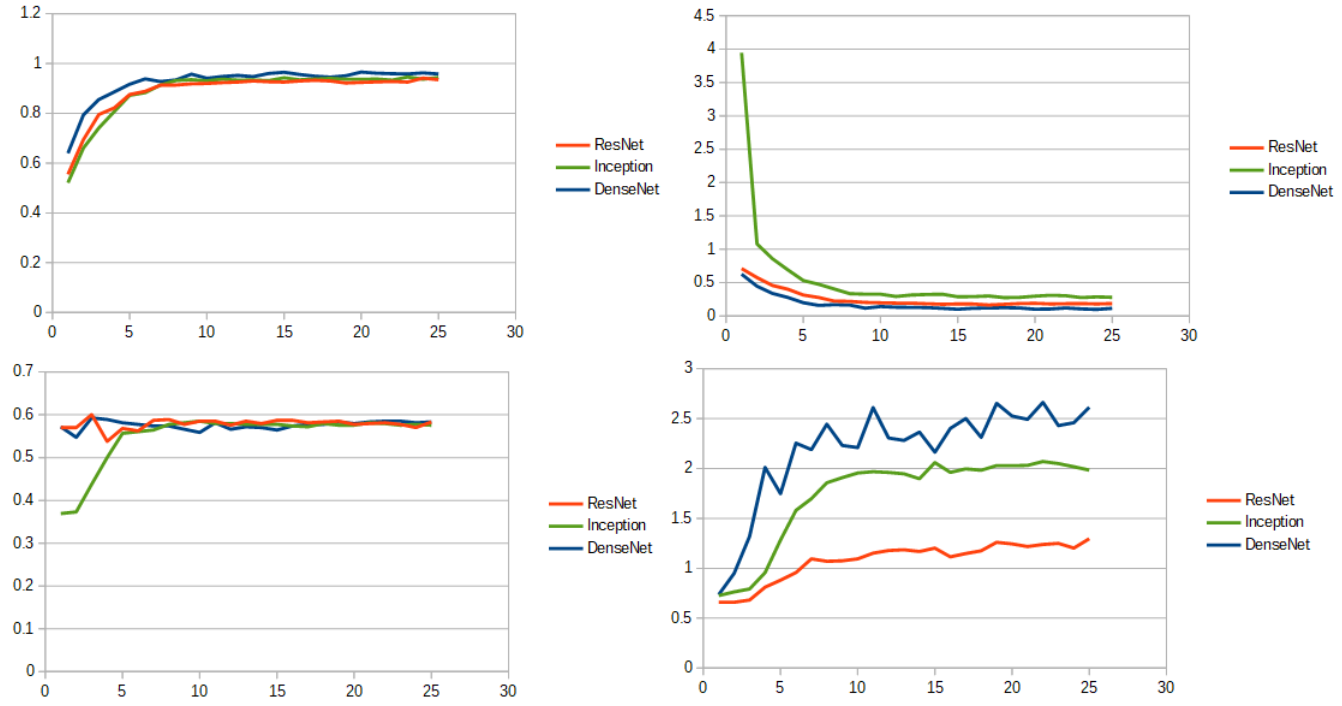
B. *ResNet*

ResNet is a deep convolutional network that exploits skip connections between layers to learn a residual function that closely models the true mapping function. This modification allows it to learn rich features representations at each layer of the model. The model we chose to use was ResNet34, which is an implementation of ResNet with 34 layers. We chose this size as it gives a balance between size and complexity [3].

C. *DenseNet*

DenseNet takes many of the same techniques implemented in ResNet, however it has a few fundamental differences. While ResNet uses a summation between layers (skip connections), DenseNet proposes a concatenation between intermediate layers, meaning that size is preserved while the

Fig. 2. Upper Left: Training Accuracy, Upper Right: Training Loss, Lower Left: Validation Accuracy, Lower Right: Validation Loss



number of features between layers grow. These areas where they concatenate layers make up ”Dense blocks” which are then put between normal convolution and average pooling layers [4].

D. Inception v3

The Inception Model was Google’s attempt to create a model for the ImageNet classification challenge. The model architecture can be seen in figure 1. It starts like most CNN’s with layers of convolution and max pooling, however the most important part of the model lies in the ”inception modules” that come after it. Stacked after the convolutions are concatenated ”inception modules” which are the meat and potatoes of the model. Each module is made up convolutions of varying filter sizes which allows it to learn features at multiple length scales. At the end of each module the filters are concatenated and passed on to the next module. To overcome the expensive big filter operations, they also make use of using 1x1 convolutions to reduce the dimensionality of the input. At the end of all the modules it is then fed through a soft max layer, which is used for classification.

VII. EXPERIMENTS

A. Experimental Setup

In order to conduct training we used GPU’s provided by Amazon. By setting up AWS accounts, we were able to use cloud computing to run our experiments. We were able to use two Tesla K80’s for training our models. Our virtual instance was running Python 3.6 and PyTorch 0.4.0.

B. Experiment Details

For each model we trained for 25 epochs using Stochastic Gradient Descent for Optimization . For hyper-parameters we use a starting learning rate of 0.001 with a momentum of 0.9. We use models pretrained on ImageNet that are then used for transfer learning to learn our Waldo dataset.

VIII. RESULTS

Results from our experiments are reported in table two. These numbers are the test accuracies of each model where the model used was the top performing model on the validation data set. In addition, figure 2 shows the loss and accuracy on both the train and validation sets during each epoch. The results in table clearly show that our models

TABLE II
TEST ACCURACY ON ALL EXPERIMENTS

	DenseNet	ResNet	Inception
Baseline	47.74%	47.74%	47.74%
Augmented	44.53%	43.58%	45.28%

trained on the augmented dataset models do worse than 50% accuracy across the board. Keeping in mind that this task is a 2 class classification task, it is doing worse than random guessing. There are many reasons that this could be the case. The first most obvious reason for this result is that the models are over-fitting to the training data. From the graph’s in figure 2, we can see that the training loss quickly decreases and training accuracy quickly approaches 100% accuracy. However, on the validation data, the loss increased across epochs while the accuracy stayed the same. This is a

tell-tale sign that our model is overfitting. We believe this is due to the way we oversampled the data which causes a very large bias in the representation that the model is learning. Another possible reason that our model performed poorly is that ImageNet is trained on real images while Where's Waldo are cartoonish illustrations and it could be possible that the features that were learned in ImageNet do not transfer over to illustrations easily (especially with the limited illustration data).

On the other hand the baseline models actually performed about the same as the augmented models. However we believe that the reasoning behind the performance is completely different. In the case of the baseline model, the data was horribly imbalanced with only 19 Waldo images in the training set. This imbalance in the training set reinforced the models on pretty much always picking "not waldo", then during test time, even when it sees a Waldo image it still calls it "not waldo". Because of the way our dataset is set up 47.74% of our test set images are "not-Waldo", so we know that (unless for some reason our model is getting very lucky), the model is in fact choosing "not waldo" every time.

We had originally chose to use Transfer Learning due to its ability to not need large amounts of data, however it seems that what the model was learning in the augmented case was not a useful representation, and it instead hampered performance.

IX. FUTURE WORK

We were unable to train models with the full augmented dataset, which had a size that would not be considered low resource at all. We hope to be able to train models on this dataset, however we are still pessimistic given our current resources. In addition, an important measure that we were unable to add in time for this submission was the inclusion of a confusion matrix on each of the test sets. This would be an important tool that we could use to see exactly how the model is making its errors and it would give further insight into our results.

In addition, we have considered another method of tackling this problem which would involve further changing of our dataset. The current positive Waldo images contain Waldo's head but also details of the background or on occasion other faces and characters. While, Waldo's face and hat are the common link between all of these training images, we believe our transfer learning models would perform better if the positive inputs provided to the models when training were only composed of Waldo's face and perhaps background but not the extra details of adjacent characters. This would require extensive scouring of Waldo puzzles to find and select Waldo (and then select negatives) to prepare such a dataset.

X. CONCLUSIONS

We show that the data augmentation technique we presented has little impact on the overall performance, and in actually lowers overall performance on the model. In addition, the models trained on the original dataset do not learn any meaningful representation, and instead fall back to just choosing the more present class. Though Transfer learning is supposed to be useful in situations data is abundant, it seems that when there is too little data, the model will still not learn. From our results, we conclude that oversampling in our case could lead to a loss in performance. The solution to this may be to generate more synthetic data (which we did but could not try out through training/testing) or to create these synthetic images from a larger set of source images by finding more Waldo puzzles and digitizing them. We mentioned some alternatives in the future work section above.

XI. POST-SCRIPT

There are a couple of things we learned by doing this project. The first was learning about setting up an AWS server for deep learning from scratch. We utilized free credits from the GitHub Student Pack which allowed us to start up deep learning server. This gave us a lot of computing power and allowed us to utilize a Tesla K80 rather than our laptop CPUs. The second thing we learned how to do was transfer learning in PyTorch. Before this assignment, we were all under the assumption that every model had to be trained from scratch in PyTorch; however, being able to leverage pretrained models allowed us to speed up our training substantially and we were able to explore different, interesting models.

Our advice to next year's DL students would be to brush up on probability, statistics, and linear algebra in order to be ready for the first part of the class. In addition, they should know the second half of the course is extremely practical and not entirely theoretical, especially when the instructors and guest lecturers talk about the variety of applications of deep learning. Our main advice to the instructors would be change the timing of the quizzes so that they don't have to be answered during lecture. We found pulled our attention away from the lectures.

REFERENCES

[1] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," *Standford Reports*, vol. 2017, 2014.

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[4] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993