Image Classification with Convolutional neural networks

*Note: Sub-titles are not captured in Xplore and should not be used

Zeduo Zhang
Master of Computer Science AI
Western University
London, Canada
zzhan762@uwo.ca

Bochen Dong
Master of Computer Science
Western University
London, Canada
bdong44@uwo.ca

***Abstract*—This project researches how to build a convolutional neural networks (CNNs) based model to classify images with multi-classes, and how to use tensorflow-gpu to reduce training time. In this research, we analyze the effects of different parts (layers, learning rate, training data, etc.) of the CNNs based model, and build a model that has a great performance on our dataset. In addition, addressing overfitting is a challenging problem during training DNNs models. We try to reduce overfitting through multiple methods: dropout, data augmentation, generative adversarial networks (GANs). Moreover, some deeper model architectures, such as ResNet and MobileNet, are tested and compared in this project. As for the environment, Python is the coding language we use and the tensorflow-gpu is the platform we use to reduce training time. Finally, we figured out that adding a dropout layer, data augmentation and using GANs to generate more new examples can really help to address the overfitting problem and increase the accuracy. Our final model is the Mobile Net as the basic architecture with data augmentation, dropout layer, decay learning rate and GANs, which can achieve a 91.23% accuracy on the test set.**

***Index Terms*—Image classification, Convolutional neural network (CNNs), Python, Tensorflow-GPU**

I. Introduction

Nowadays, image classification is widely developed and used in various technically advanced fields: medical imaging, satellite object identification, machine vision, etc [2]. It is a complicated process that can easily be influenced by bias or tricks. To overcome this issue, multifarious techniques and theories were developed, and DNNs is the most reliable, promising and deeply discussed approach. Convolutional neural networks (CNNs) based models, a class of DNNs, achieve good results in several critical conditions. CNN plays a significant role in image classification because it can detect smaller patterns relative to the whole image. The pattern of an object can appear anywhere in the image multiple times, which can be recognized by the convolution layer in CNN. Meanwhile, we know that subsampling the image would not affect the identification of the object and thus we can subsample the image to reduce the processing time, which is the function of max pooling layer in CNN [3].

Overfitting is also a serious problem during training DNNs. For complicated dataset, a simple model architecture may not have the ability to solve the problem, that is the generalization ability is weak. However, a model with too much capacity would overfit the training dataset [4]. In addition, the learning rate is also a significant factor about which we need to be concerned during the model training process. A low learning rate takes a long time to train the model while a high learning rate may not approach the optimal solution and cause an unstable learning process. In this project, we would not study the learning rate in detail, we take a moderate initial learning rate and a dynamically decreasing learning rate like learning rate decay or learning rate reduction to train our model.

Therefore, it is important for a junior learner of machine learning to know how CNNs work, what are the effects of different parts in CNN and how to use CNNs to solve basic image classification problems. In this project, we use the dataset from Kaggle website with 150 classes, and try to train a competitive CNNs based model that has good classification performance on the dataset.

Tensorflow is an end-to-end platform for machine learning, created by the Google Brain team [5]. It provides comprehensive and useful tools, library and dataset, etc. for programmers to implement machine learning projects. Using tensorflow-gpu, we can load the dataset to GPU, and preprocess the dataset and train the model on GPU which saves a lot of training time. Since our dataset has 7,000 images with 150 classes, and the deeper architectures we tried require lots of time to train, we decided to use tensorflow-gpu to run our project.

II. Dataset

Our dataset is from the Kaggle website [1], maintained and owned by Lance Zhang. Most of the images from this dataset are imported from another dataset on Kaggle [6]. Lance Zhang went through all these images, and modified them so that all images are labelled correctly and applicable for machine learning. Based on the description from the website, there are 7,000 labeled and hand-cropped Pokemon images with 150 classes of which the class names are also the name of

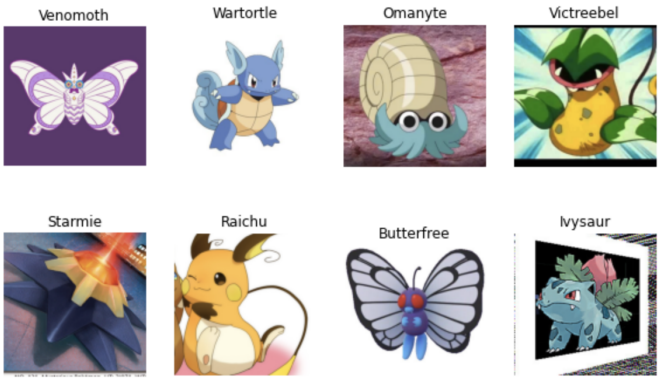

Fig. 1. examples from dataset

pokemons. In addition, each class contains 25-50 images with the pokemon in the center. Most of them have relatively high quality and all perfect for light deep learning. Before loading these datas to our model, we do some preprocessing on the dataset. we resize all the images to (192, 192, 3). Moreover, passing large pixel values to the model causes the process to be complex, and thus we normalize all the pixel values to range from 0 to 1. According to Fig. 2 which is the plot of count of images, we can see that some classes have more than 60 images, but some only have about 25, less than the half of the largest. We can see that the number of images among different classes are unbalanced. But based on the cases mentioned in [7], it is not a problem in this case. However, we will try to fix this using generative adversarial neural network (GAN) which would be discussed below in detail.

For the images in each class, we split them into three sets: 70% for the training set, 10% for the validation set, 20% for the test set. Training set and validation set are used for training models, and test sets are used to evaluate our models.

Fig. 1 shows some examples from the dataset.

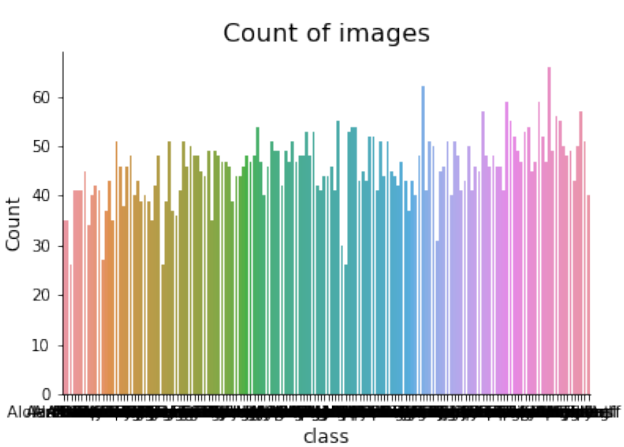

Fig. 2. Count of the number of images in each class

III. METHODOLOGY

Since the data we have is image data and the CNN (convolution neural network) performs well on the image data. So we decide to construct a CNN model to fit on the training data.

A. First attempt: Base Model

In this section of work, we constructed a base model which has 3 convolution layers with 3×3 filter size and 3 fully connected layers. We use Adam as our optimizer with learning rate equals to 0.001, and we define the cross entropy loss as our loss function.

Since the the number of images for each class is relatively low, therefore, in the first place, we believe a simple neural network is sufficient for us to have a good performance on both training set and validation set.

After trained by 30 epochs, the training accuracy of the base model converges to 98% and the validation accuracy converges to 35%. The training loss keep decreasing and the validation loss decrease from 5 to 3.3 between epochs 0 and 4 then increase from 3.3 to 8. It is clearly for us too see that the validation accuracy is poor and the over-fitting happened badly after epoch 4.

From the above result, we believe the model is memorizing the data and it understand poorly about the meaning and connection between the images and their name. There are serval ways to overcome this problem: 1) *obtain more training data from the original training set.* 2) *change to a better model that can better interpret the connection between images and labels*

B. Data Argumentation

Data augmentation is a very powerful and useful technique or process that greatly improves the performance of the model by creating new examples from existing training data. It can help to improve the diversity and complexity of the training dataset, and consequently address the overfitting problem. For this part, we will compare several keras data augmentation techniques [8], and analyze their effects on our training dataset and model.

What we will discuss here are RandomFlip, RandomRotation, RandomContrast, RandomZoom and RandomTranslation. The view of effect of them are shown in Fig. 5.

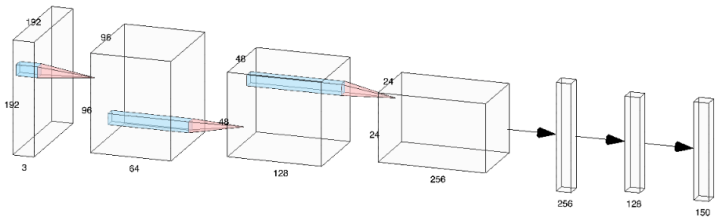

Fig. 3. The Base Model is a 6-layer network, which include 3 convolution layers and 3 fully connected layers.

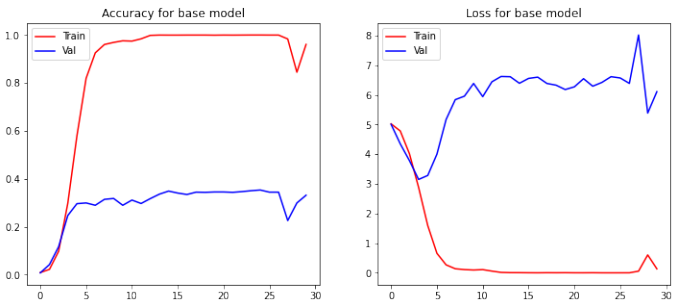

Fig. 4. The accuracy and loss on training set and validation set. The training accuracy converges to 98% and the validation accuracy converges to 35%. The training loss keep decreasing and the validation loss decrease between epochs 0 and 5 then increase.

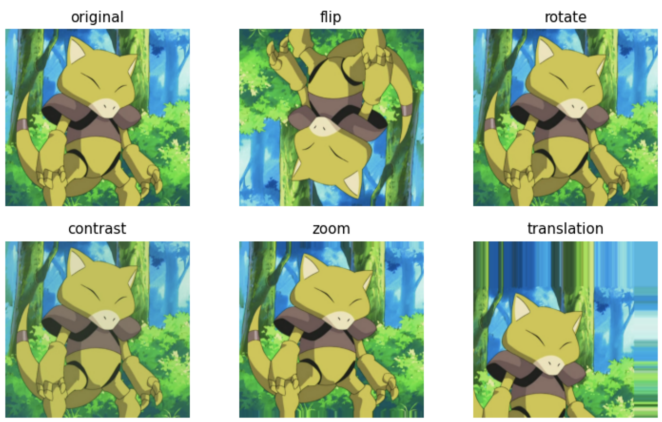

Fig. 5. views of each data augmentation techniques

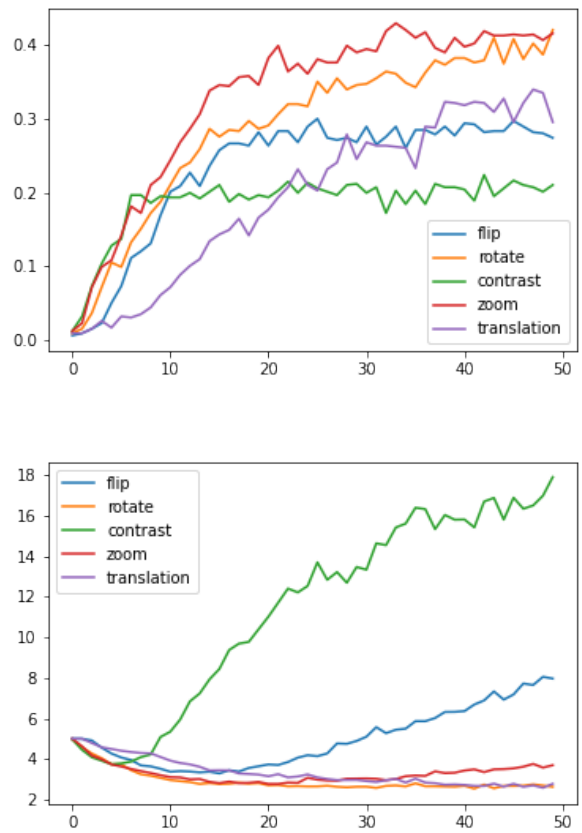

Fig. 6. the validation accuracy and loss of the baseline model with different data augmentation techniques

- RandomFlip: Randomly flip the image horizontally or vertically.
- RandomRotation: Randomly rotate the image within a specified range
- RandomContrast: Adjust the contrast of the image with a random factor
- RandomZoom: Randomly zoom in or zoom out the image with a random factor
- RandomTranslation: Randomly shift the image vertically and horizontally within a specified range and then the points outside the boundaries of the input are filled according to the given mode, and for this project, we choose the "nearest" mode which is also applied for the RandomRotate and RandomZoom techniques.

In Fig. 6, we compare different data augmentation techniques. FFrom the validation accuracy graph, we can see that RandomZoom, RandomRotation and RandomTranslation have relatively high accuracy while RandomContrast has the worst performance. Based on the validation loss graph, RandomRotation and RandomTranslation are the best because they have the lowest and decreasing loss function. RandomZoom has a relatively low and stable loss function, but its validation loss of zoom is kind of slightly going up after 20 epochs.

Based on the information above, we have a basic idea of the order of the effectiveness of each technique. and follow this order, we wanna take a look at the effectiveness of the combination of them. We choose RandomZoom as our first choice, because it has the highest validation accuracy. Although its validation loss is worse than RandomRotation and RandomTranslation, it is not a big problem because there is just a mild increase after 20 epochs. Based on the validation accuracy and loss graph Fig. 7, the combination of RandomZoom and RandomRotation has the highest accuracy but lowest and decreasing loss function.

Finally, Fig. 8 can apparently demonstrate that the data augmentation we chose greatly address the overfitting problem.

C. Dropout

Dropout is the layer that will randomly ignore some of the neuron during steps of training, that is the effect of this neuron is ignored during forward, and the weight would not be modified during backward. That is dropout can make a neuron not too sensitive to weight change and thus improve generalization ability and address overfitting problems. In our project, we will discuss and analyze three different approaches to add the Dropout layer [9].

- Adding the dropout layer after each Max Pooling layer.

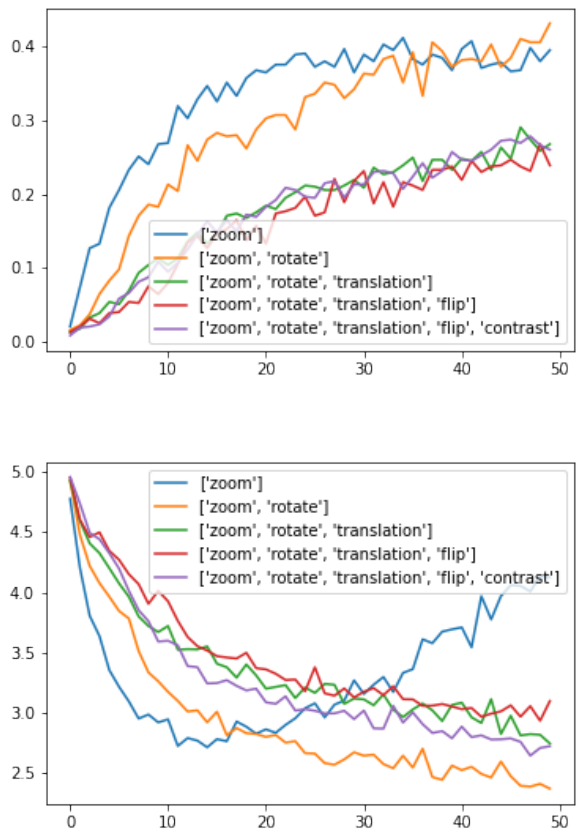

Fig. 7. the validation accuracy and loss of the baseline model with different combination of data augmentation techniques

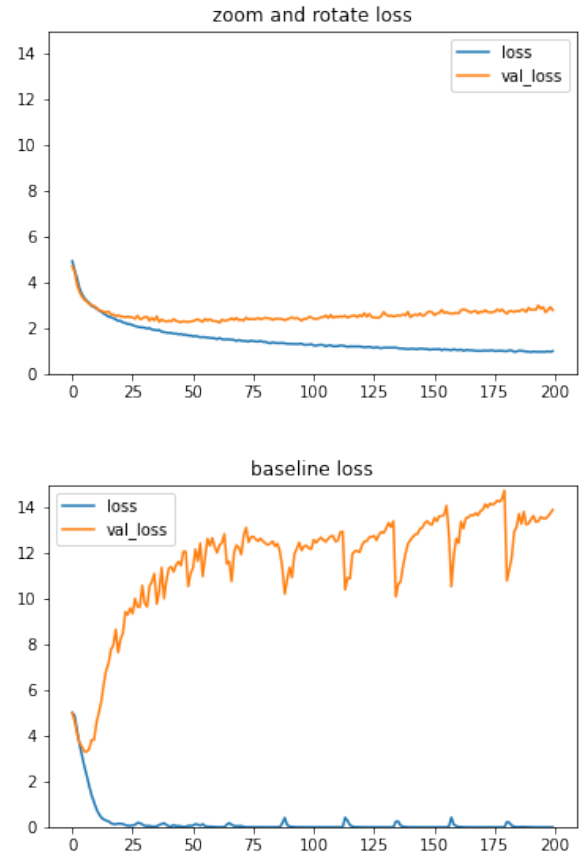

Fig. 8. The top one is the training loss and validation loss of the baseline model using RandomZoom and RandomRotation. The bottom one is the trainling loss and validation loss of baseline model.

- Adding spatial dropout layer after each convolutional layer.
- Adding dropout layer between visible layers.

Fig. 9 shows the result of the first choice, and we use the baseline model with a different dropout rate (0.2, 0.4, 0.6) to test the difference. From the accuracy plot, we can see that with the rate increased, the accuracy is also increasing. And from the validation loss plot, with the rate increase, the overfitting problem is addressed. So we can see that when the rate is 0.6, the performance of the model is the best

The result of the second choice shown in Fig. 10 reveals that adding a spatial dropout layer after each convolutional layer seems not effective and suitable. With the accuracy increased, the loss function instead gets worse, which means the overfitting problem is more serious. That is there is a kind of tradeoff between accuracy and loss, and thus it is hard to choose an appropriate dropout rate. But compared with the baseline model, it still can help to address the overfitting problem.

Fig. 11 shows the result of adding the dropout layer between visible layers. We can see that the trend is really similar with the first choice, and this choice can really help to address the overfitting problem. But different from the first method, a smaller rate is better. So when the rate is 0.2, the performance is the best.

Finally, we try to combine these three methods mentioned above, for example, in Fig. 12, we add dropout layers with rate 0.6 after each Max Pooling layer, and then add dropout layers between visible layers with rate (0.2, 0.4, 0.6). We can see that the blue line which is the model with dropout layers after each Max Pooling layer has the best performance. The problem may be that adding too many dropout layers causes the neural network to ignore too many neurons and consequently affect the performance.

From the results shown above, we can see that adding appropriate dropout layers with an appropriate rate can help to address the overfitting problem. In addition, where to add the dropout layers, how many dropout layers we should add and what is the dropout rate we should use are the problems needed to discuss and analyze.

D. GAN

https://www.tensorflow.org/tutorials/generative/dcgan

After applying Data augmentation to our model, we checked our data set, we found out approximately 20% of our class

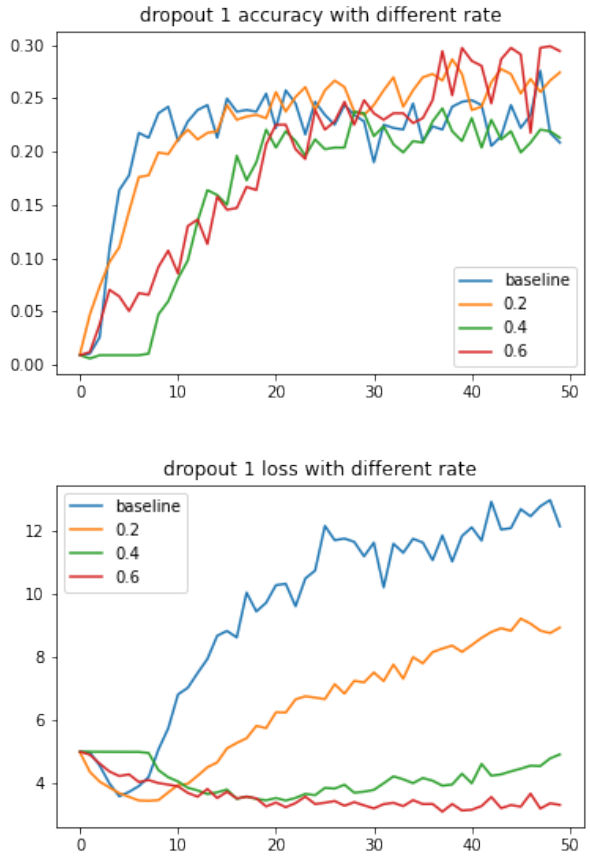

Fig. 9. the top one is the validation accuracy of baseline model and baseline model with different dropout rate after each Max Pooling layer. the bottom one is the validation loss

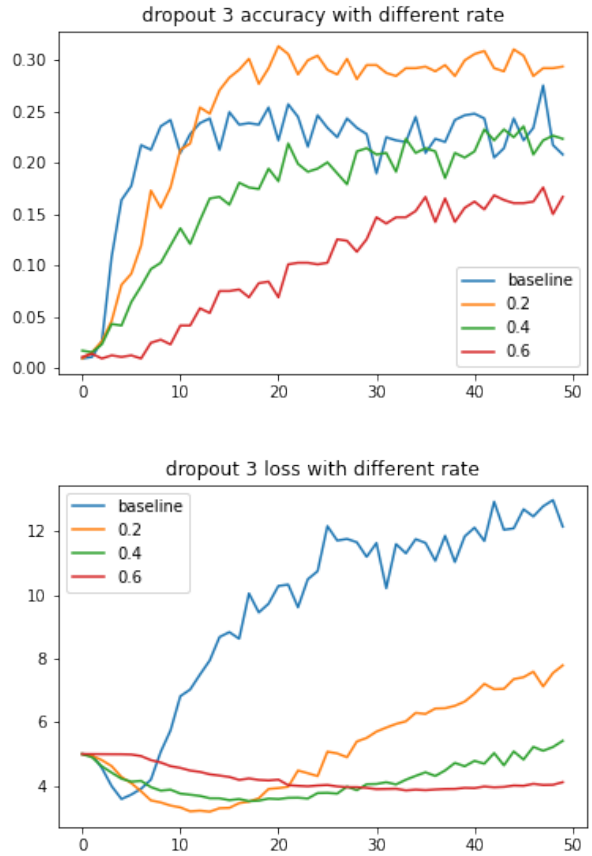

Fig. 10. the top one is the validation accuracy of baseline model and baseline model with different spatial dropout rate after each convolutional layer. the bottom one is the validation loss

only has 30 images or below, we then take a guess, since the number of training images for some classes is too low, this may be the reason that limits our model to have better accuracy.

In this section of work, we propose a pipeline described in figure above that works on the original training set without an additional data collection process. First we use a network structure called DCGAN [11] run 4000 epochs to generate 36 fake images for each class, these fake images are labeled as same as the class label. Then we saved the fake image generated by the generator to our training data. Finally we will run the base model on the new training data and compare the loss and accuracy on both training set and validation set, this idea was inspire by Unlabeled Samples Generated by GAN Improve the Person Re-identification Baseline in vitro [12].

The main challenge we meet in this section is:

1) *How to use the newly generated data:*
   As the image shown above, We have 36 new images generated by the generator each epoch, are we going to use all of them, if not, which set of fake images should we add to the training data.
2) *How to obtain more training data from the original training set:*
   The fake images we got are similar to one or two specific images in the original data set. On the one hand, we can add regularization method to reduce the similarity of the images generated by DCGAN. On the other hand, we can use a smaller set of training images to generate fake images to reduce the similarity.
3) *The computing power.*
   We use one Ge-force 1070 GPU to generate fake images, it takes 30 minutes for each class to generate 36 new images. Therefore, if we want to generate fake images for 150 classes, it will cost us 75 hours, which is unacceptable for us.

As a result, the new data set generated by GAN does not have a significant improvement. As we can see in the figure above, our model converge very fast and achieve 98% accuracy on the training set. However, the accuracy on the validation set is still low. We believe GAN can achieve a better performance after we overcome above challenges, and we will do this in the future work.

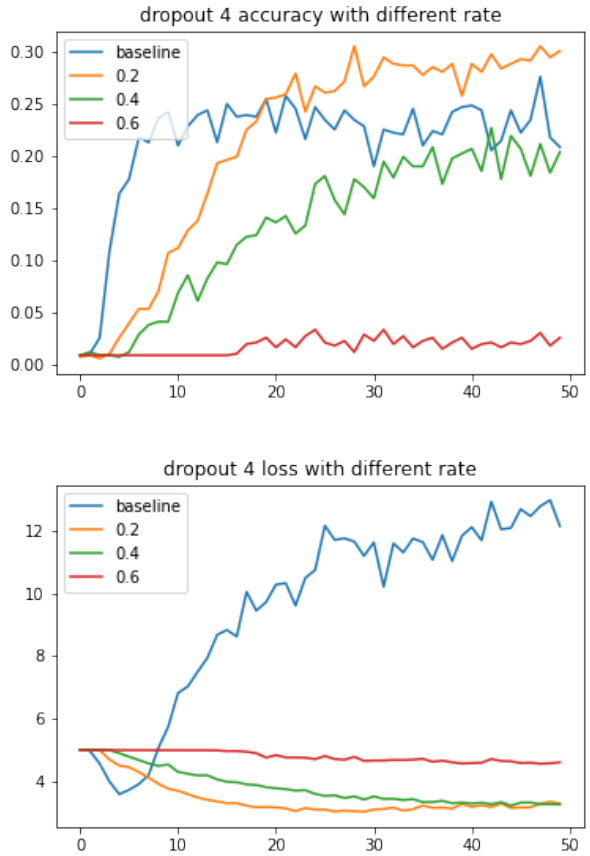

Fig. 11. the top one is the validation accuracy of baseline model and baseline model with different dropout rate between each visible layers. the bottom one is the validation loss

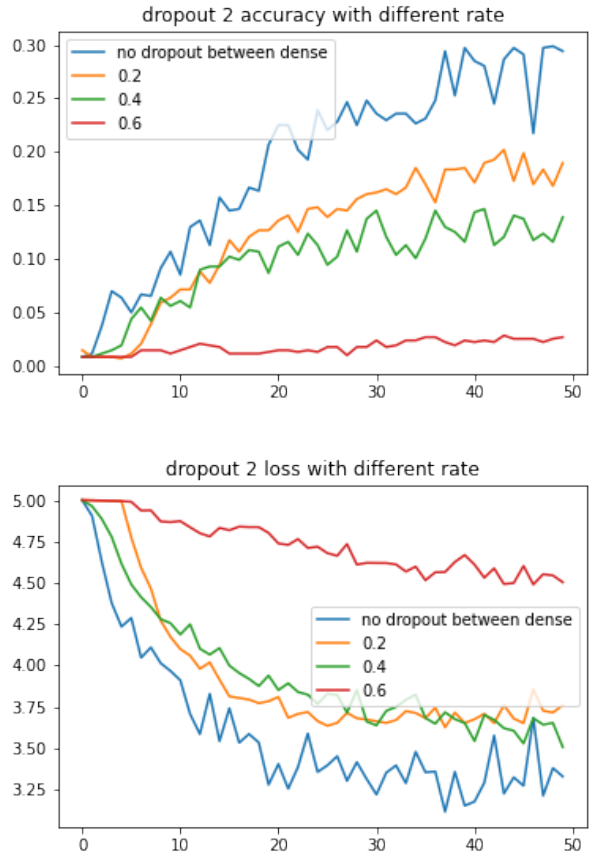

Fig. 12. The blue line stands for the baseline model with dropout layers after each Max Pooling layer. The rest of the lines are the baseline model with dropout layers after each Max Pooling layer and dropout layers between each visible layers with rate indicated in the legend. The top one is the validation accuracy while the bottom one is the validation loss

E. Going to a deeper network

After we made these attempts, we begin to consider why the validation accuracy is still low. we thought it might be that the complexity of the model itself is too low, therefore when the base model tries to fit a more complex classification problem, it can only memorize the data and understand poorly about the meaning and connection between the images and their label. Therefore, we decided to try a deeper network structure to see if it can have a better performance. We tried VGG16 in the first place, and we use different learning rates and different loss functions, but the VGG does not perform well on a small dataset. So we then considered resnet50.

1) ResNet [13]: We all know that as the model going deeper and deeper, vanishing gradient problem and gradient exploding problem are likely to occur. There are two main reason that we choose ResNet50 in this section:

1) *Bottleneck architecture*:
   ResNet50 solves vanishing gradient problem and gradient exploding problem by using bottleneck architecture. We can see there are 3 layers in the bottleneck architecture. The three layers are 1X1, 3X3, and 1X1 convolutions, where the 1X1 layers are responsible for reducing and then increasing dimensions, leaving the 3X3 layer a bottleneck with smaller input/ output dimensions. [13]
2) *Residual Learning*:
   Instead of hoping each few stacked layers directly fit a desired underlying mapping, ResNet50 let these layers fit a residual mapping. If denoting the desired underlying mapping as $H(x)$, we let the stacked nonlinear layers fit another mapping of $F(x) := H(x) - x$. The original mapping is recast into $F(x)+x$. Optimizing the residual mapping is easier than optimizing the original mapping. [13]

Because of these advantages of ResNet50, we decide to train a ResNet50 model on our training set in this section to see how will it performs.

Now let's look at the loss and accuracy on the training and validation set. We can see that under the same learning rate. Since ResNet50 did not converge after 30 epochs, therefore we trained the ResNet50 for 100 epochs. As we can see in the below figure, ResNet50 converges much slower than the base model. And we can see that the ResNet50 has a significant improvement in validation accuracy. This encourages us to

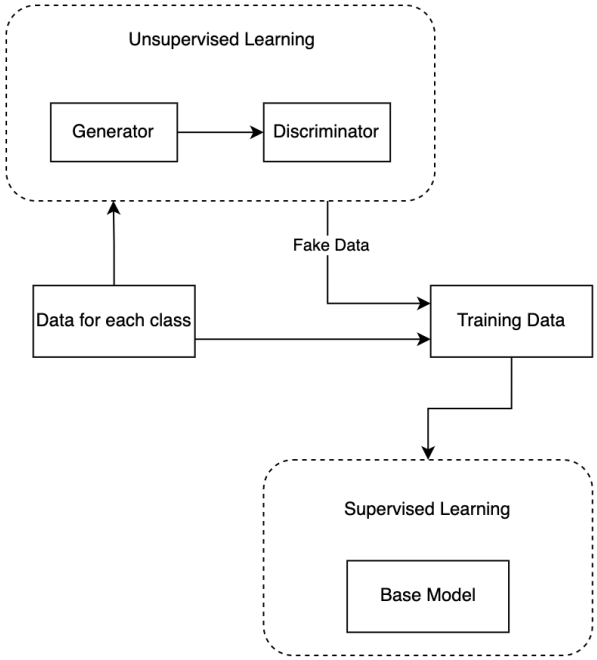

Fig. 13. The pipeline of the proposed method, run 4000 epochs on DCGAN to generate 36 fake images for each class. Training data includes the real and fake images for each class.

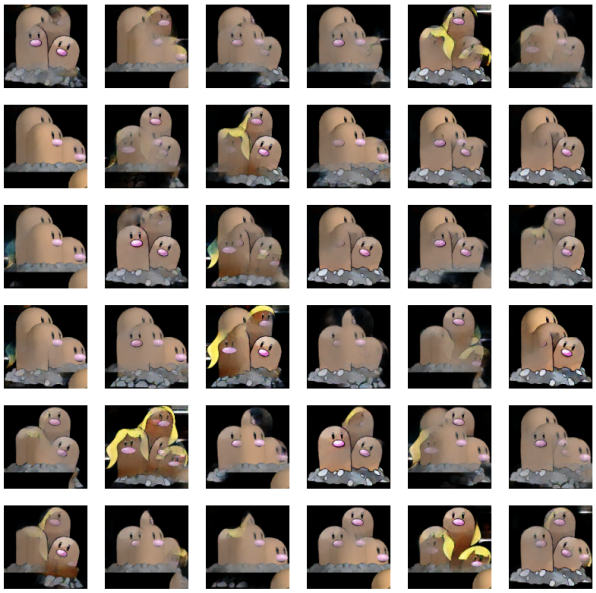

Fig. 14. 36 fake images generated by DCGAN after 4000 epochs.

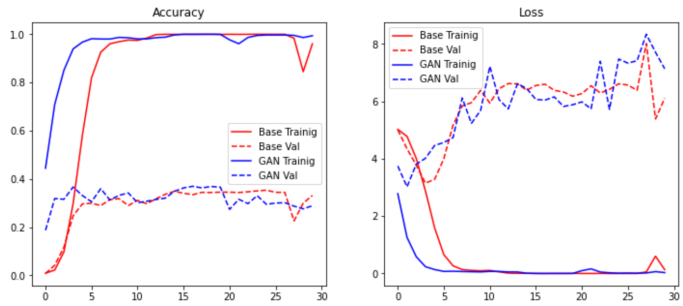

Fig. 15. The accuracy and loss on training set and validation set. Under the same learning rate, GAN converges faster than the base model and it has a higher validation accuracy between epoch 15 and 20. However, this improvement is not that significant.

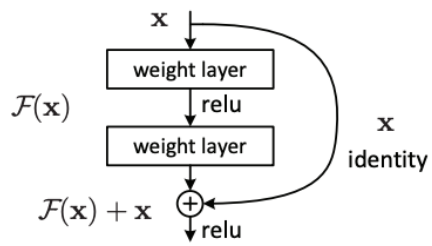

Fig. 16. Residual Learning: a building block

explore how deeper models will perform.

2) MobileNet [14]: In this section of work, we tried the mobile net version 2 on the training set.

The mobile net version 2 [15] has its architecture based on an inverted residual structure where the input and output of the residual block are thin bottleneck layers. We can see from the picture above; the original residual block has a higher dimension on the input and output layer compared to the

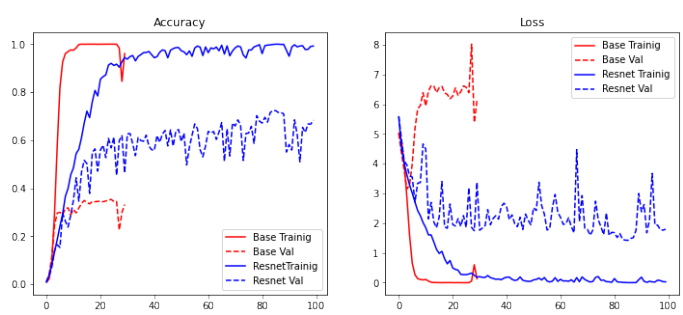

Fig. 17. Residual Learning: a building block

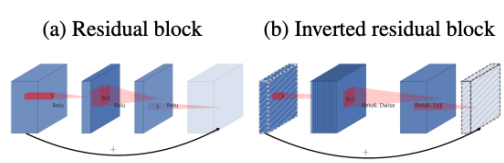

Fig. 18. Inverted residual structure

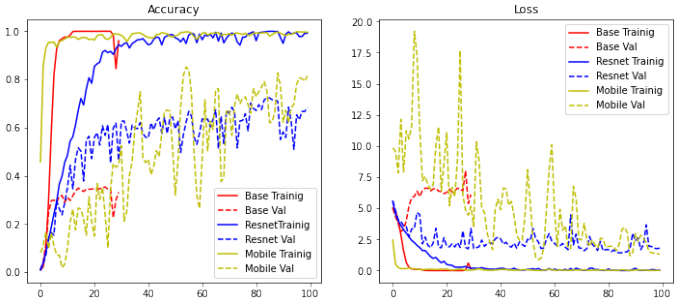

Fig. 19. Residual Learning: a building block

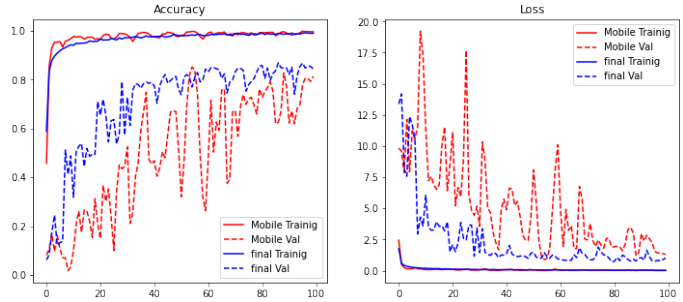

Fig. 20. The accuracy and loss of Final Model with data augmentation training on the data-set who contains real images and fake images generated by DCGAN.

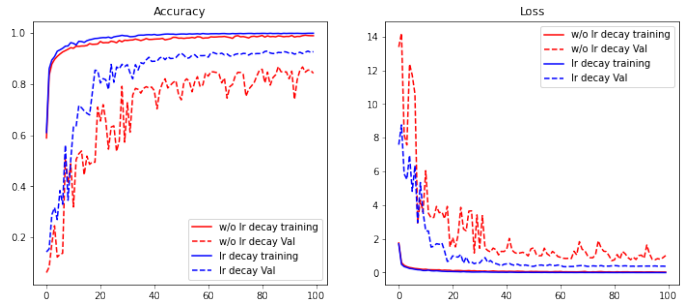

Fig. 21. The accuracy and loss of Final Model with and without learning rate decay. b

converted residual block. The mobile net also uses a linear transformation to replace the ReLU in the layer with a small number of channels.

Then let's look at the loss and accuracy on the training and validation set. We can see that under the same learning rate, Mobile-Net converges much faster on the training set than the base model and ResNet50. It also has the best validation accuracy among these three models. But it is very unstable, so we need to apply learning rate decay or learning rate reduction in the future work.

F. Final Model

In this section we will train a mobile-net and apply the techniques we talked about above. Since the data augmentation has a significant improvement, we will apply random rotation and random zoom in our model. Then as we described in the section above, Mobile net is very unstable during training, this may because the learning rate is relatively large, Therefore we will try learning rate decay after data augmentation.

We first tried data augmentation first on the on the data-set contains real images and fake images generated by DCGAN. The result is shown in figure 18. We can see under the same learning rate, final model has a better accuracy on validation set and it is much more stable than the previous model. This also reflect on the loss, the final model also has a lower loss on validation set.

We then apply learning rate decay with initial learning rate equals to 0.001, decay rate equals to 0.98 and decay steps equals to 200. After applying learning rate decay to our model, we can see the model with learning rate deacy is much more stable than the original model and it increase the validation accuracy to around 90%.

IV. Conclusion

In conclusion, our project shows the basic knowledge of using CNNs to do image classification. We study and analyze different techniques used to address the overfitting problem which is common in machine learning, and verify that adding dropout layers, data augmentation can reduce the overfitting. Especially, when the number of examples in each class are unbalanced or number of examples in a specific class is extremely low, using GAN to generate more new examples can help to increase the accuracy and address the overfitting problem. In addition, choosing a model with deeper architecture can increase the capacity of the model to perform well, however, at the same time, the deeper model may cause overfitting which is greatly addressed by approaches mentioned above. For our final model, we choose Mobile Net as our basic architecture, use GAN to generate more examples to our training dataset, and then add dropout layers, data augmentation contains RandomZoom and RandomRotation. This model has great performance and achieves 91.23% accuracy on the test dataset. However, our GAN has a horrible performance which is not expected, because it always produces the same image from the original training dataset. That is it just memorizes and does not generate, and suffers mode collapse [10]. Therefore, in the future, we need to improve our GAN model to generate new and useful images, and test the data augmentation, dropout layers and learning rate to choose the most capable one. In addition, we will try to analyze the effect of regularization which is another technique to address the overfitting problem.

Acknowledgment

I would like to acknowledge Professor, Charles Ling, for inspiring our interest and providing us the basic and state-of-the-art knowledge on machine learning through well structured course schedule and attractive course delivery style.

References

[1] L. Zhang, "7,000 labeled Pokemon," Kaggle, 09-Nov-2019. [Online]. Available:

https://www.kaggle.com/lantian773030/pokemonclassification. [Accessed: 15-Dec-2021].

[2] viso.ai. 2021. A Complete Guide to Image Classification in 2021 - viso.ai. [online] Available: https://viso.ai/computer-vision/image-classification/. [Accessed 15 December 2021].

[3] Ling, C., 2021. CNNs.

[4] Brownlee, J., 2021. How to Avoid Overfitting in Deep Learning Neural Networks. [online] Machine Learning Mastery. Available: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/. [Accessed 15 December 2021].

[5] Yegulalp, S., 2021. What is TensorFlow? The machine learning library explained. [online] InfoWorld. Available: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html. [Accessed 15 December 2021].

[6] Kaggle.com. 2021. Pokemon Generation One. [online] Available: https://www.kaggle.com/thedagger/pokemon-generation-one. [Accessed 15 December 2021].

[7] Chatterjee, S., 2021. Deep learning unbalanced training data?Solve it like this.. [online] Medium. Available: https://towardsdatascience.com/deep-learning-unbalanced-training-data-solve-it-like-this-6c528e9efea6. [Accessed 15 December 2021].

[8] TensorFlow. 2021. Module: tf.keras.layers |TensorFlow Core v2.7.0. [online] Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers. [Accessed 15 December 2021].

[9] Brownlee, J., 2020. How to Reduce Overfitting With Dropout Regularization in Keras. [online] Machine Learning Mastery. Available at: ¡https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/¿ [Accessed 15 December 2021].

[10] C. Wang, C Xu, X Yao, D Tao, "Evolutionary Generative Adversarial Networks", IEEE Transactions on Evolutionary Computation, vol. 23, no. 6, Dec., pp.921-934, 2019.

[11] Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative Adversarial Networks. arXiv.org. arXiv preprint arXiv:1701.07717, 2016.

[12] Zheng, Z., Zheng, L., & Yang, Y. Unlabeled samples generated by Gan improve the person re-identification baseline in vitro. arXiv preprint arXiv:1701.07717, 2017.

[13] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 , 2015.

[14] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., amp; Adam, H. MobileNets: Efficient convolutional neural networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861 , 2017.

[15] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., amp; Chen, L.-C. MobileNetV2: Inverted residuals and linear bottlenecks. arXiv preprint arXiv:1801.04381 , 2019.