Title: Who's the Pokemon?

G018 (s1821025, s1846832, s1836909)

Abstract

This project aims to develop an efficient drawn-images classification program for Pokemon characters based on convolutional neural networks (CNNs) technology. It employs three model architectures, including PlainNet, ResNets and DenseNets, and compare their performance on image classification. Our data set consists of 1000 64*64 images for each of 100 Pokemon characters, resulting in 100,000 pictures totally. All images are downloaded from Google Pictures by Microsoft Azure. The results present that ResNet-101 model achieves the highest accuracy score (69.54%) in recognizing Pokemon characters.

1. Introduction

Machine-learning technology nowadays has been applied to many aspects of our lives, from searche engines to commercial webpage recommendations. Specifically, machine-learning systems are used in image objects identifications, speech-text translations, and products recommendations based on users' interests (LeCun et al., 2015). In recent years, it is increasingly used in smartphones and cameras applications.

One limitation of traditional machine-learning techniques is processing natural data in raw form(LeCun et al., 2015). Therefore, numerous methods and algorithms have been discussed for decades in order to solve these artificial intelligence vision problems, such as image classification and object detection(Guo et al., 2016). Deep learning technology, as a state-of-the-art method with multiple levels of representation(LeCun et al., 2015), now plays an important role in training neural networks and solving computer vision tasks. Increasingly, applications make use of it to solve natural data. For instance, face recognition technology is increasingly used in smartphones like Apple. As most deep learning algorithms are currently focus on processing natural data and have performed successfully, we are curious about how it performs in drawn-images object classification.

As a simple and most representative image classification experiment, MNIST digits recognition task shows current-best human-approaching performance with an error rate less than 0.3%(Krizhevsky et al., 2012). However, unlike simple handwritten digits recognition, the complexity of classifying other objects in the real world can be much immense.

Hence, this project examine the performance of state-of-art machine learning approaches on relatively more complex objects classification. Our research of classification is based on drawn Pokemon images. The comic Pokemon, as a memory of our childhood, is one of the most popular topics since we were in primary school. It has released a list of animations, games, and surrounding, etc. The game "Pokemon Go" now is also one of the most favourite games. According to analysis, by 2018 this AR game has included around 380 million active users around the world, and this data is keeping an up-going trend.

In plot setting of Pokemon, there are around eight hundred of different Pokemon characters. For each Pokemon character, people can easily get access to its images with different postures and forms through the Internet. However, it may be not easy to identify every Pokemon character sometimes when we come across interesting Pokemon photos or peripheral dolls. Therefore, this research develops a deep learning model which improves the accuracy of classifying Pokemon characters.

Compared to standard neural networks with similarly-sized layers, convolutional neural networks (CNNs) can make strong and mostly correct assumptions about the nature of images nature using much fewer layers and parameters (Krizhevsky et al., 2012). It is easier to train. In this coursework, we aim to use three kinds of CNNs to do image classification, including PlainNet, ResNets and DenseNets.

As a kind of standard deep neural network architecture, Residual networks (or ResNets) show state-of-the-art performance across various applications. With the output of one layer is directly added to the output of some following layer (Shamir, 2018), the connection between layers of this kind of networks is skipped.

DenseNet is a typical method to alleviate the redundancy in CNNs. Unlike networks copying features from early layers in traditional layer-by-layer connectivity pattern, DenseNet connects each layer with all layers before it. This reduces feature replication (Iandola et al., 2014).

We aim to implement the the three types of CNNs introduced above to the experiment of Pokemon drawn-images classification and figure out the best model for characters' identification. In this report, following data composition and task in the next section, we will introduce the two CNNs in the third section in detail. We will show the experiments and results in the following two section and then conclude.

2. Data set and task

2.1. Data

Our data set includes 100 Pokemon characters. For each class, we collected around 1000 images for each class from Google Pictures by Microsoft Azure. We reshaped the size of those images to 64 * 64, and their colour channel is three. The data set is transformed by Torchvision package in python. Each class is named by a certain number (from 1 to 100). We divide the data set to train set, validation set and test set (8:1:1). We do preprocessing to the data set - delete irrelevant images and make the main part of the image show Pokemon's figure by cropping the image. Some portraits and cosplay photos are also included in the dataset to explore the practicality of the model in the real environment.

One thing that should be considered is, the colour of the image largely affects the performance of the model. The structures and colours of Pokemon characters are not too complicated. Thus, the noise of images is not a big challenge for our implementation. On the other hand, even some different Pokemon characters may have very similar features. For example, Pikachu and Raichu were created with mice as a source of inspiration. We may find they have a very similar appearance if ignoring their colours. As a result, three-channel input is necessary, although it greatly lengthens the training time.

2.2. Task

For recent years, the application of Deep learning has brought up an improvement in image classification. However, many architecture models are based on natural images. We could not evaluate the model's performance on drawn images. Pokemon is a very popular comic in our young ages. There are a large number of Pokemon characters that can be applied to evaluate the performance of neural networks in processing drawn images. We will be possible to create a real "Pokemon chart" during this implementation. Our main task is to apply different model architectures on the Pokemon data set. Based on the performance of the model in predicting the test set, we select the best model architecture to do the Pokemon classification.

3. Methodology

Deep learning enables machines to achieve image recognition and speech recognition with very high accuracy through a deep network. The most direct way of improving the accuracy of the model is to design the network as deep as possible. However, the reality we are facing is that the performance of the model gets worse after too many layers are directly stacked on the plain network (the exact number depends on the specific architecture). The error results of the training set and the test set are shown in figure 1. The effect is getting worse (the error rate goes higher) when the network is getting deeper (from 20 layers to 56 layers). This shows that the model becomes more difficult to train

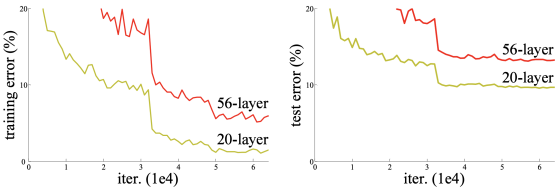

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer plain networks. The deeper network has higher training error, and thus test error.(He et al., 2016)

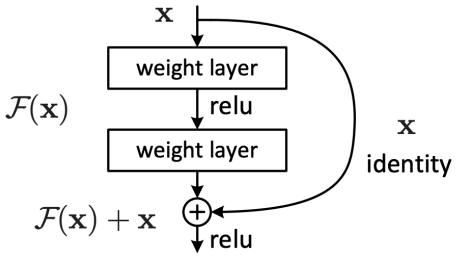

Figure 2. Residual learning: a building block.(He et al., 2016)

when the network becomes deep. The neural network continuously propagates the gradient during backpropagation with a certain degree of loss. When the number of network layers is too large, the gradient gradually approaches zero during the propagation process, which makes it difficult to effectively adjust the weight of the previous network layers.

3.1. The architectures of ResNet and DenseNet

Deep residual network. The emergence of Deep Residual Network (ResNet) solves the problem we mentioned above (the effect is getting worse when the network is getting deeper). ResNet introduces a residual network structure. Under this network architecture, the network depth can be greatly increased, and the final classification effect is also excellent. The basic structure of the residual network is shown in the figure 2. The structure is with a jump structure. Suppose that the input of a particular neural network is x, and the expected output is F(x), that is, F(x) is the expected complex potential mapping. The task is to learn such a model.

In the figure 2 above (the figure is from (He et al., 2016)), through the "shortcut connections" method, the input x is directly passed to the output, and the output result is $H(x) = F(x) + x$. When $F(x) = 0$, then $H(x) = x$, which is so called the identity map. Therefore, the ResNet is equivalent to changing the learning target, instead of learning a complete output, but the difference between the target value $H(X)$ and x, which is called the residual $F(x) := H(x) - x$. Therefore, the latter training goal is to approximate the residual result to 0 so that the accuracy does not decrease as the network

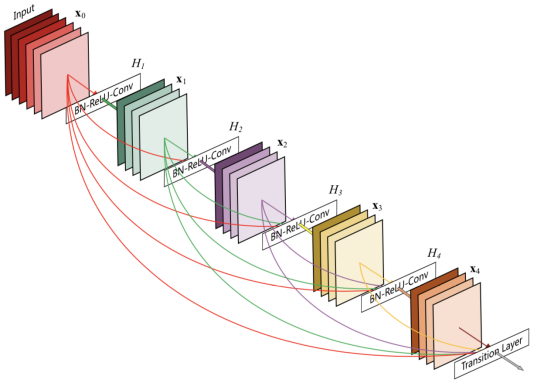

Figure 3. A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input.(Huang et al., 2017)

.

deepens. This kind of residual jump structure breaks the convention that the output of #n-1 layer can only be the input of #n layer so that the output of a certain layer can directly cross several layers to be the input of the latter layer. The significance of the ResNets is that it provides a new direction for solving the "deeper neural networks make the error rate of the model goes up instead" problem.

Densely connected convolutional network. Compared to ResNet, densely connected convolutional network (DenseNet) (Huang et al., 2017) introduces a more aggressive connecting mechanism: to interconnect all layers (shown in figur 3). Specifically, each layer accepts all of its previous layers' outputs as its additional input. ResNet has a short-circuit connection between each layer and a previous (generally two or three) layer. In DenseNet, each layer is concatenated with all previous layers (where the size of feature maps of different layers is the same) and used as the input to the next layer. This is a dense connection and can achieve feature reuse, compared to ResNet. This feature is the main difference between DenseNet and ResNet. To express it with a formula, the output of a traditional network at the layer is:

$$x_l = H_l(x_{l-1}) \tag{1}$$

For ResNet, the identity function from the previous input is:

$$x_l = H_l(x_{l-1}) + x_{l-1} \tag{2}$$

In DenseNet, all the previous layers are connected as the input:

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{3}$$

Among them, the equation (3) above represents a non-linear transformation, which is a combined operation, may include a series of Batch Normalization (BN), ReLU, Pooling and Convolution operations.

The regular CNN generally needs to reduce the size of the feature map by a pooling layer or a convolution layer (stride>1), whereas the densely connection method of DenseNet requires the feature map size to be consistent. To solve this problem, the "dense block and transition" structure is used in the DenseNet architecture. The dense block is a module with many layers. The feature map size of each layer is consistent and the layers are densely connected. The Transition module connects two adjacent dense blocks and reduces the size of the feature map by pooling. Figure 4 shows the network structure of DenseNet, which contains three dense blocks, and each dense block is connected by Transition.

3.2. The methodology of evaluating models

Based on the experience, we know that the factors affecting the performance of CNNs are: the network structure and the value of hyperparameters. Thus we will evaluate the best performing architecture in terms of these two factors. Before evaluating these two types of networks mentioned above, we first train a baseline model and the performance of that is seen as a benchmark. The Baseline model is built with the Plain Network.

Plain network. First we train a plain network to be the baseline model, which is a benchmark of our experiments.

Residual network. Based on the Plain Network, we insert shortcut connections to train ResNet model. When training ResNet, we use the value of hyperparameters (batch size, learning rate, weight decay, momentum) in (He et al., 2016). We do not use dropout. In terms of network structure, we evaluate various network structures according to the different settings of network layers. We use the performance of the model in processing the validation set as a criterion for evaluating network performance. To avoid accidental results, each architecture is trained for three times and the average results is determined to be the final results.

Densely connected convolutional network. Next, we train the densely connected convolutional network. In terms of hyperparameters, we use the values of the hyperparameters (learning rate, weight decay, momentum, grow rate) in (Huang et al., 2017). We add a dropout layer after each convolutional layer (except the first one). In terms of network structure, we evaluate various network structures according to the different settings of network layers. The mechanism for selecting the network structure of the best performance is the same as above.

Finally, we compare the performance of all network structures. We use the best performing network structure as the optimal model of the Pokemon classification and use the accuracy in predicting the test set as the final result of this task.

4. Experiments

4.1. Detailed architectures

First, we determine the detailed network structures that will be used.

Plain Network. We train a 4-layer and 5-layer plain network. The architecture are composed of 3*3 convolution

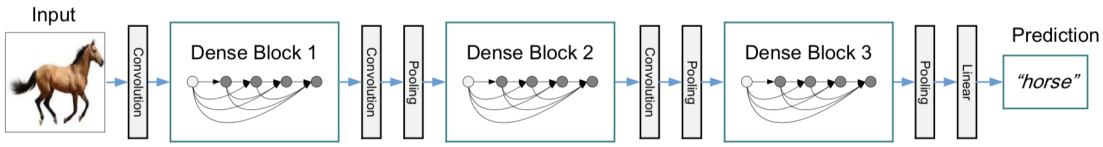

Figure 4. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.(Huang et al., 2017)

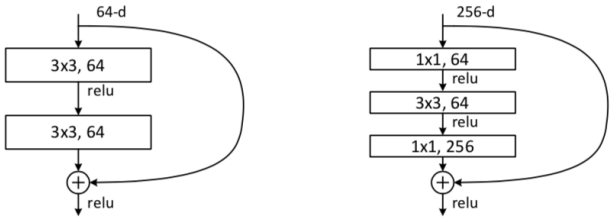

Figure 5. A deeper residual function F for Pokemon. Left: a basic block (on 64*64 feature maps) for ResNet - 34. Right: a "bottleneck" building block for ResNet-50/101/152.(He et al., 2016)

layers and 2*2 max-pooling layers, followed by a 2*2 adaptive average-pooling layer at the end of the last max-pooling layer. The subsampling is performed by max-pooling with a stride of 2. The detailed network structure is shown in Table 1. We use the learning rate of 0.1, a weight decay of 0.0001, and batch size of 100.

Residual Network. We built ResNet with 34, 50, 101 and 152 layers, respectively. Each architecture consists of a different number of blocks (shown in Figure 2). We use the same hyperparameter value (learning rate = 0.1, weight decay = 0.0001, momentum = 0.9) as in the (He et al., 2016).

34-layer ResNet. The 34-layer network consists of a 3*3 convolution layer, 16 basic blocks (figure 5 left), and an average-pooling layer. The network ends with a 1000-way fully-connected layer, and softmax (Table 2).

50-layer ResNet, 101-layer ResNet, and 152-layer ResNet. On the basis of the 34-layer network, we replaced each basic block with a bottleneck block (Figure 5 right) to build a 50-layer network. We continue to increase the number of bottleneck blocks to build 101-layer and 152-layer networks (Table 2).

Densely connected convolutional network. We built DenseNet with 121 layers, 169 layers and 201 layers. Each architecture consists of dense blocks (Figure 3) and transition layers (Table 3). We use the same hyperparameter value (learning rate = 0.1, weight decay = 0.0001, momentum = 0.9, growth rate=32, reduction=0.5) as in the (Huang et al., 2017).

121-layer, 169-layer, and 201-layer DenseNet. In our experiment, each framework contains four dense blocks and three transition layers. Before entering the first dense block, a convolution with 64 (or twice the growth rate) output channels is performed on the input images. Each dense block contains several 1*1 convolutions and 3*3 convolutions. Then, the transition layer containing a 1*1 convolution and a 2*2 average pooling is performed. At the end of the last dense block, a global average pooling is performed and then a softmax classifier is attached. Subsampling of all architectures is performed by the 2*2 average-pooling within the transition layer (Table 2). The feature-map sizes in the four dense blocks are 64 * 64, 32 * 32, 16 * 16, and 8 * 8, Respectively.

4.2. Evaluate neural networks.

Our data set consists of 100 Pokemon characters, each containing 500 64*64 images. The ratio of the size of the training set, validation set, and test set is 8:1:1. We test the models on these data sets first. To make the final model have higher accuracy in the actual scene (using the mobile phone camera to recognize Pokemon), we collect 500 more images for each Pokemon character so that each class has a more adequate data set. In this way, our new data set consists of 100 Pokemon characters, each containing 1000 64*64 images. Then we divide the data set into training set, verification set, and test set according to the ratio of 8:1:1. We apply the data set to model architectures above to train our Pokemon classifier. Finally, we choose the best model based on the accuracy in predicting test data.

5. Results and discussion

Based on our experiments, the baseline model (plain network) put up a disappointing result; ResNet-101 achieved the highest accuracy on the test set. Even though we have increased the number of layer of plain network, the final accuracy is still unreliable. Not only four layers but also five layers are not sufficient for such a 100-class task. Due to the small size of our images and the structure of the plain network, simply increasing the number of network layers may not help improve the model's performance. The fitting process of ResNet and DenseNet is shown in figure 6 and figure 7. For both ResNet and DesNet, their upper limit of the number of layers is higher than plain network. In our experiments, the best model is ResNet-101. For Res-152, more network layers do not lead to better performance, in

Layer Name	Output Size	Filter
Conv 1	64*64	3*3, 64
Max-pooling 1	32*32	2*2, stride 2
Conv 2	32*32	3*3, 64
Max-pooling 2	16*16	2*2, stride 2
Conv 3	16*16	3*3, 64
Max-pooling 3	8*8	2*2, stride 2
Conv 4	8*8	3*3, 64
Max-pooling 4	4*4	2*2, stride 2
Adaptive Average-pooling	2*2	2*2, stride 2
	1*1	Fully-connected, Softmax

Table 1. Paint Network structure for Pokemon classification.This is a 4-layer architecture. Down-sampling is performed by max-pooling 1, max-pooling 2, max-pooling 3, and max-pooling 4 with a stride of 2.

Layer Name	Output Size	34-layer	50-layer	101-layer	152-layer
Conv 1	64*64	3*3, 64, stride 1			
Conv 2_x	64*64	$\begin{bmatrix} 3*3, 64 \\ 3*3, 64 \end{bmatrix}$ *3	$\begin{bmatrix} 1*1, 64 \\ 3*3, 64 \\ 1*1, 256 \end{bmatrix}$ *3	$\begin{bmatrix} 1*1, 64 \\ 3*3, 64 \\ 1*1, 256 \end{bmatrix}$ *3	$\begin{bmatrix} 1*1, 64 \\ 3*3, 64 \\ 1*1, 256 \end{bmatrix}$ *3
Conv 3_x	32*32	$\begin{bmatrix} 3*3, 128 \\ 3*3, 128 \end{bmatrix}$ *4	$\begin{bmatrix} 1*1, 128 \\ 3*3, 128 \\ 1*1, 512 \end{bmatrix}$ *4	$\begin{bmatrix} 1*1, 128 \\ 3*3, 128 \\ 1*1, 512 \end{bmatrix}$ *4	$\begin{bmatrix} 1*1, 128 \\ 3*3, 128 \\ 1*1, 512 \end{bmatrix}$ *8
Conv 4_x	16*16	$\begin{bmatrix} 3*3, 256 \\ 3*3, 256 \end{bmatrix}$ *6	$\begin{bmatrix} 1*1, 256 \\ 3*3, 256 \\ 1*1, 1024 \end{bmatrix}$ *6	$\begin{bmatrix} 1*1, 256 \\ 3*3, 256 \\ 1*1, 1024 \end{bmatrix}$ *23	$\begin{bmatrix} 1*1, 256 \\ 3*3, 256 \\ 1*1, 1024 \end{bmatrix}$ *36
Conv 5_x	8*8	$\begin{bmatrix} 3*3, 512 \\ 3*3, 512 \end{bmatrix}$ *3	$\begin{bmatrix} 1*1, 512 \\ 3*3, 512 \\ 1*1, 2048 \end{bmatrix}$ *3	$\begin{bmatrix} 1*1, 512 \\ 3*3, 512 \\ 1*1, 2048 \end{bmatrix}$ *3	$\begin{bmatrix} 1*1, 512 \\ 3*3, 512 \\ 1*1, 2048 \end{bmatrix}$ *3
	1*1	Average-pooling, 1000-d fc, Softmax			

Table 2. ResNet architectures for Pokemon, with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

terms of our Pokemon classification task.

In our experiments, DenseNet does not reach the highest accuracy. Table 4 shows the details of each model. The difference in accuracy between DenseNet-201 and ResNet-152 is not significant. However, the number of parameters of DenseNet-201 is significantly smaller than that of ResNet-152 (2×10^7 parameters and 6×10^7parameters respectively), which will greatly reduce the training time and calculation cost of the model. Compared to ResNet, DenseNet is not too complicated. Our DenseNet models can optimise the classification results by increasing the growth rate to use more convolution kernels. In this task, all DenseNet's growth rate (K) is 32. When processing Cifar-10, (Huang et al., 2017) used DenseNet-169 with K = 48 to achieve similar accuracy with ResNet-152. Whereas the number of parameters of the model has doubled, Compared to DenseNet-169 with K=32. Doing so increases training time and computational cost significantly. So we give up on this model.

The accuracy of the test set is the only factor of evaluating different model architectures. In the case of small differences in terms of accuracy, space consumption and time consumption are also worth considering.

Even though we have tried to delete those irrelevant and same images, some of those not be found may be a reason that our model cannot perform like expected.

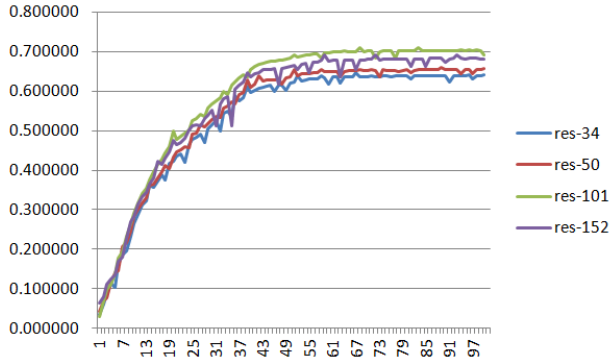

Figure 6. Validation accuracy using four ResNet architectures.

Layers	Output Size	121-layer	169-layer	201-layer
Convolution	64 * 64	3 * 3 conv, stride 1		
Dense Block (1)	64 * 64	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 6$	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 6$	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 6$
Transition Layer (1)	64 * 64	1*1 conv		
	32 * 32	2 * 2 average pool, stride 2		
Dense Block (2)	32 * 32	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 12$	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 12$	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 12$
Transition Layer (2)	32 * 32	1 * 1 conv		
	16 * 16	2 * 2 average pool, stride 2		
Dense Block (3)	16 * 16	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 24$	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 32$	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 48$
Transition Layer (3)	16 * 16	1 * 1 conv		
	8 * 8	2 * 2 average pool, stride 2		
Dense Block (4)	8 * 8	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 16$	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 32$	$\begin{bmatrix} 1*1conv \\ 3*3conv \end{bmatrix} * 32$
Classification Layer	1 * 1	average pool, 1000-d fc, softmax		

Table 3. DenseNet architectures for ImageNet. The growth rate for all the networks is k = 32. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

Model Architecture	Train	Val	Test	Better?
PlainNet_{4layer}	NULL(<0.1)	NULL(<0.1)	NULL(<0.1)	
PlainNet_{5layer}	0.2173±0.0021	0.1947±0.0172	0.1905±0.0235	
ResNet-34	0.7657±0.0023	0.6404±0.0045	0.6439±0.0039	
ResNet-50	0.7768±0.0020	0.6579±0.0035	0.6237±0.0044	
ResNet-101	0.8168±0.0013	0.7013±0.0031	0.6954±0.0027	√
ResNet-152	0.8153±0.0017	0.6849±0.0024	0.6793±0.0027	
DenseNet-121	0.7972±0.0022	0.6714±0.0013	0.6411±0.0031	
DenseNet-169	0.7856±0.0031	0.6574±0.0259	0.6323±0.0024	
DenseNet-201	0.7221±0.0117	0.6872±0.0344	0.6814±0.0231	

Table 4. Training accuracy, validation accuracy, and test accuracy on Pokemon data set. Both "train" and "validation" are values from the last epoch. Each value is derived from the mean and standard deviation of the three sets of experiments trained for each architecture.

6. Conclusion

Deep learning methods currently have performed well in natural image classification, however, there is few research using this technology in drawn-image classification. This project explores how convolutional neural networks (CNNs) technology perform in Pokemon drawn-images.

Specifically, three CNNs model architectures, including PlainNet, ResNets and DenseNets, are employed in this research. In our tests, 100,000 64*64 images are used for training, testing and validating models. Some portraits and cosplay photos are also included in the dataset to explore the practicality of the model in the real environment.

According to our experiment results, Deep network shows well performance in Pokemon drawn-image classification tasks. The highest accuracy score we achieve is 69.54% based on ResNet-101 model. Moreover, the performance of ResNet and DenseNet architectures are better than PlainNet. In the end, we determine Res-101 to be our Pokemon classifier.

References

Guo, Yanming, Liu, Yu, Oerlemans, Ard, Lao, Songyang, Wu, Song, and Lew, Michael S. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

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.

Huang, Gao, Liu, Zhuang, Van Der Maaten, Laurens, and Weinberger, Kilian Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Iandola, Forrest, Moskewicz, Matt, Karayev, Sergey, Girshick, Ross, Darrell, Trevor, and Keutzer, Kurt. Densenet:

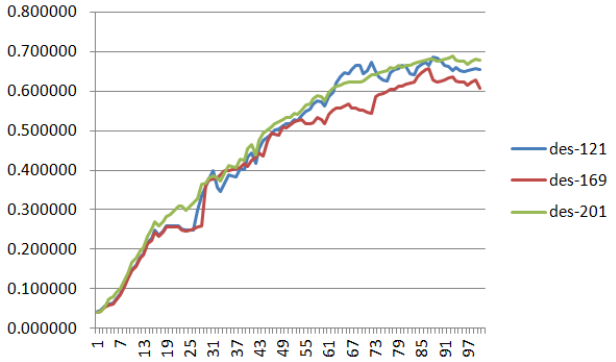

Figure 7. Validation accuracy using three DenseNet architectures.

Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *nature*, 521(7553):436, 2015.

Shamir, Ohad. Are resnets provably better than linear predictors? *arXiv preprint arXiv:1804.06739*, 2018.