**Pets Breed Identification**

Group25

Yingjin Song u6122877

Weitao Chen u6309924

Jinming Dong u6683369

Zirui Chen u6227730

**Abstract**

As one of the specific applications in the convolutional neural network, pet identification is considered as a fine-grained image classification problem [4]. To solve this challenge, we aim to implement and train from three state-of-art models, VGG16, ResNet50 and Xception to perform breed classification on The Oxford-IIIT Pet Dataset [5]. In order to reduce the overfitting that can be caused by the complex breed identification problems, we explore and apply several techniques, such as Data Augmentation, Dropout, Batch Normalization, etc. The final classification model can categorize the cat or dog category among the 37 categories given any cat or dog image with top-1 accuracy of 89.9% and top-5 accuracy of 99.8%.

**1 Introduction**

Convolutional Neural Networks (CNN) have been widely applied and gained remarkable success in image classification and object recognition, which profits from the powerful computational capacity of GPU and the availability of massive image datasets represented by ImageNet [6]. In this project, the objective is to build a classification model capable of identifying the cat or dog category given any cat or dog image. To train and evaluate the model, we use The Oxford-IIIT Pet Dataset [5] containing 25 dog breed categories and 12 cat breed categories with roughly 200 images for each class.

We realize several difficulties in implementing the pet breed classification model. Firstly, the images in the dataset have large variations in scale, pose, background and lighting. Secondly, there are a variety of pet breeds, and many of the breeds can have subtle differences in appearance between other breeds. It’s not only hard for human brain to identify, but also difficult to imagine the specific features that neural network must learn in order to correctly classify these cats and dogs. Thirdly, the dataset is relatively small in terms of the number of images for each category and also the resolution for each image.

In this project, we build our CNN models using TensorFlow and MXNet both operated in Python primarily. We explore and compare the architectures including VGG16, ResNet50 and Xception. We also experiment with the techniques that can potentially improve the performance including Data Augmentation, Dropout, Batch Normalization. By comparing these results, we evaluate the impact of different models, parameters and methods. At last, we apply the best performing model to classify the external pet images.

**2 Related Work**

Similar to the pet breed classification, a lot of researches have been conducted to solve the simpler problem of distinguishing images of cats and dogs into two categories. Elson et al. [7] achieved the 56.9% test accuracy on the ASIRRA dataset by using the model based on the color features. Golle et al [8] got the accuracy 82.7% by using Support Vector Machines (SVM) classifier based on the combination of features, color and texture, which are extracted from the images. Parkhi et al. [5] applied the Scale-Invariant Feature Transform (SIFT) features to train a classifier and obtained the accuracy of 92.9% finally.

To tackle the dog breed classification problem, Liu [9] and his team proposed a localization and classification pipeline using part localization. They built exemplar-based geometric and appearance models of dog breeds and their face parts, which got the recognition rate of 67% on the dataset with 133 dog breeds. Furthermore, Wang et al. [10] achieved the 20% improvement by using the model of the shape features as points on the Grassmann manifold on the basis of statistical methods for landmark-based shape representation.

**3 Architectures**

To solve this challenge, three CNN architectures, including VGG16, ResNet50 and Xception, are explored and implemented.

**3.1 VGG16**

VGG16 is one of the structures in VGG, and the other one is named VGG19. There is no significant difference between them, only the depth of the network. In order to better describe the VGG16, it is essential to introduce the AlexNet first, which is proposed by Krizhevsky et al. in 2012[16]. VGG16 is based on AlexNet but deeper and wider. AlexNet is a neural network with 8 layers, including 5 convolutional layers which may be followed by max-pooling layer, 3 fully-connected layers. Besides, it uses ReLu activation function rather than Sigmoid or tanh and uses SGD as its optimizer. What is more, AlexNet uses 3 different sizes of convolution kernel, which are 11 \* 11, 5 \* 5 and 3 \* 3. The structure of AlexNet is shown in Fig.1.

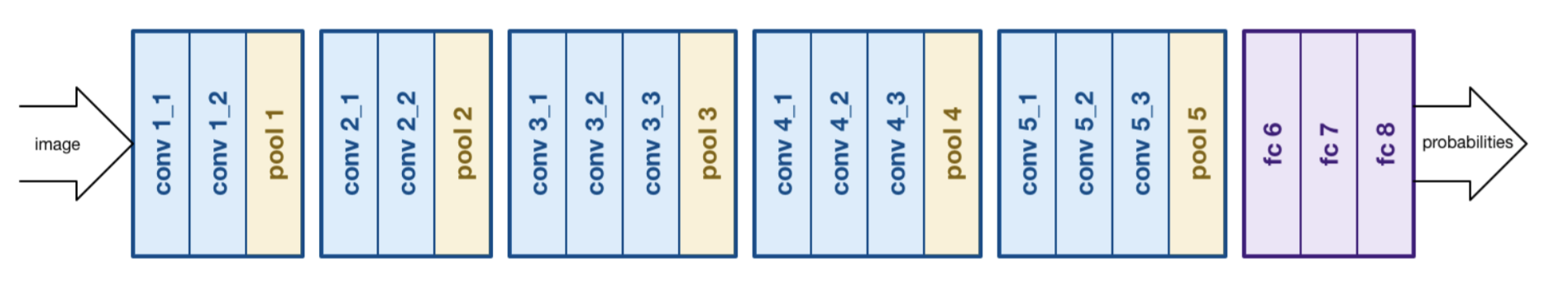
图片包含 文字, 地图

描述已自动生成

*Fig. 1: AlexNet structure by Siddharth Das (2017)*

*Retrieved from: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5*

Based on AlexNet, VGG16 makes some improvement. Firstly, unlike AlexNet, it uses 16 layers, and this is because more layers can complete more complicated tasks and achieve higher accuracy. Secondly, VGG16 replaces the large convolution kernel by piling up several 3 \* 3 convolution kernel. There are several advantages for using small convolution kernel size. For example, it can reduce the number of parameters and multiple non-linear layers can increase the depth of neural network with a lower cost [17]. Besides, VGG16 uses 2 \* 2 pooling kernel but AlexNet uses 3 \* 3 pooling kernel. The structure of VGG16 is shown in Fig.2.



*Fig. 2: VGG16 structure by Siddharth Das (2017)*

*Retrieved from: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5*

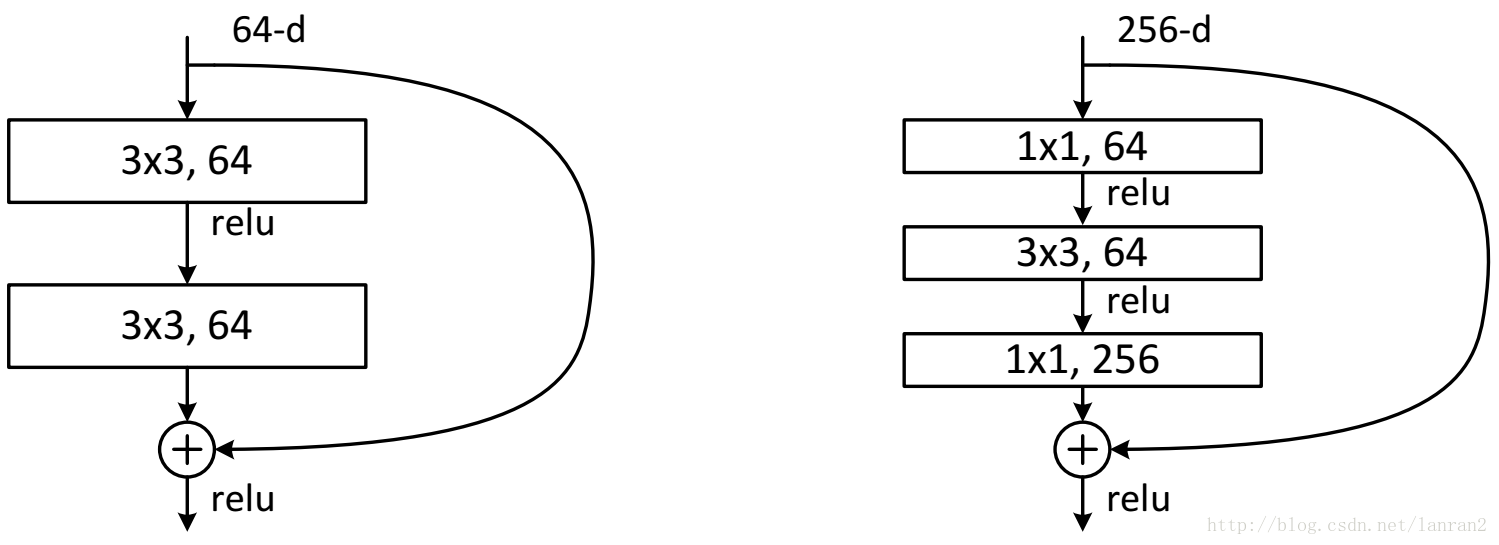
**3.2 ResNet50**

On account of previous work on neural networks, people might deduct that the depth of the networks is a critical feature that related to the performance of the networks. However, with the increasing of the depth of the neural network, it also causes problems.

One of the main problems is the vanishing gradient [13]. Vanishing gradient problem is a problem recognized in neural network training which relates to gradient-based learning methods and backpropagation. In the process of training, the curve of subsequent layers is decreasing at the process of propagating backwards, therefore later to a point that it is vanished. And it can totally stop the further training of this neural network in some situations.

In ResNet, it uses the skip-connections technique to help address the vanishing gradient problem. During the process of backpropagation, the skip connection’s pass can pass the gradient update. Instead of waiting for gradient to propagate back one layer at a time, skip connection’s path allow gradient to reach those beginning nodes with greater magnitude by skipping some layers in between [14]. Hence, Resnet can get almost the same result with other approaches, also fastens the training process of the model.

In this project, we specifically look at ResNet50. Compare with ResNet34, and this model replaces each 2-layer block with 3-layer bottleneck block that uses 1x1 convolutions to reduce then restore the depth (shown in Fig. 3). In this way, it allows for a reduced load when calculating the 3x3 convolution.



*Fig. 3: ResNet34 2-layer Block (LEFT) and ResNet50 3-layer Block (RIGHT)*

*Retrieved from: https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cc5d0adf648e*

**3.3 Xception**

Based on the ideas of Inception architecture, Chollet [1] proposed a more lightweight network named Xception in 2017, which with the smallest weight serialization at only 91MB.

In a conventional convolutional network, the convolution layer would look for cross-space and cross-depth correlations at the same time. But in the Inception, Szegedy et al. [2] separated the image region and channels slightly by using the convolutions of 1 by 1 to project the original input into smaller and isolated spaces. And for each of these input spaces, they applied a different type of filter to perform the transformation on the smaller 3D modules of the data. Instead of just dividing the input data into several compressed blocks, spatial correlations were mapped separately for each output channel in the Xception, and then depthwise convolution of 1 by 1 was performed to obtain cross-channel correlations. See Fig.4.

A close up of text on a white background

Description automatically generated

*Fig. 4: Illustration of depthwise separable convolution in Xception model*

*Retrieved from https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41*

The Xception hypothesizes that “the cross-channel correlations and spatial correlations are entirely separable, and it is best not to map them jointly”. It modifies the main idea of the original depthwise separable convolution with two differences: Firstly, Xception starts with the 1x1 convolution, and then the channel-wise spatial convolution. Secondly, Xception doesn’t contain a non-linear activation function of ReLU after the 1x1 convolution, which is included in the original Inception module. There are 36 convolutional layers totally for the base of feature extraction in the Xception architecture, in which entry flow contains 8 convolutional layers, middle flow contains 24 and exit flow contains 4 layers.

**4 Methodology**

In practice, we realize the overfitting can be a problem that decreases the performance of the model. Hence, we explore and apply data augmentation, dropout and batch normalization these methods to hopefully comfort the overfitting to some extent.

**4.1 Data augmentation**

As we all know, in the neural network, it is not possible to gain a good performance model with small dataset and data augmentation is an important method, which is able to solve this problem. Data augmentation can reduce overfitting and get better performance on the neural network, as it could generate more data from the original data by image translating, shifting, scaling etc. And there are two different types in data augmentation, one is offline augmentation [15], using for small data set, the other one is online augmentation, using for online dataset. Essentially, data augmentation is to produce the same object with other patterns that can still be recognize by human beings, and we want computers to do the same thing. Fig. 5 shows the results of data augmentation.

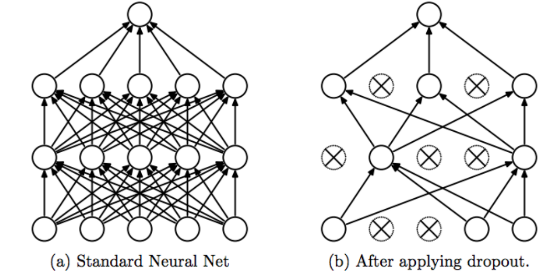
*Fig. 5: Examples of data augmentation by scaling, rotation and so on.*

In CNN, objects with different orientations, sizes or illuminations etc., which can be done by data augmentation, will not affect its results which means that CNN is invariant to these. Another function of data augmentation is that it can reduce the irrelevant features of the images in the dataset like flipping the images horizontally, this is quite useful in training CNN model and increase the accuracy of it.

**4.2 Dropout**

Dropout is a common technique that people used to reduce overfitting in neural networks by breaking up co-adaptations on training data to make the model more robust and averaging the model. Dropout refers to ignoring units during the process of training a neural network model. The simplified process of Dropout in neural network is shown in Fig. 6.

This technique effectively forces the network to stop specializing in learning features from particular the input dataset but learn more robust and generalized features that could appear in other datasets [12].



*Fig.6: Standard Neural Net VS. NN after Dropout.*

*Retrieved from: http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf*

**4.3 Batch Normalization**

During the process of training the traditional deep neural network, along with the continuous renewal of parameters, the data distribution of each intermediate layer changes quite substantially, which leads to the network need to adapt constantly to the new data distribution increasing the difficulty of training. We can only solve this problem with using a relatively small learning rate and finely-tuned initialization parameters. And the deeper the intermediate layers are, the more noticeable this phenomenon is. The phenomenon is referred to as Internal Covariance Shift.

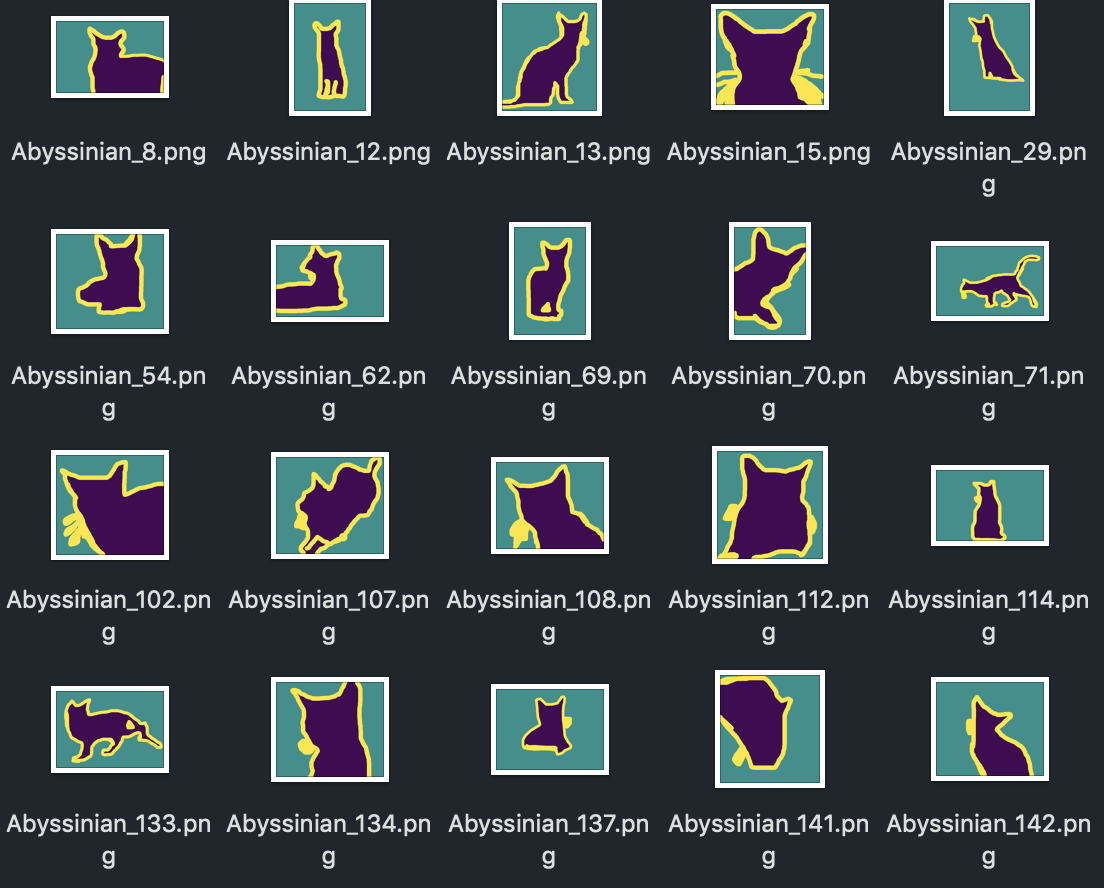
To solve this problem, Ioffe and Szegedy (2015) [11] proposed the idea of Batch Normalization (BN). The concept is to standardize not only the input layer, but also the input of each intermediate layer before activation function by calculating the mean and variance value of a small batch of data at the current layer. It results to the output is normally distributed with the mean of 0 and the variance of 1.

For the pet breed identification problem, because of the complexity, the extracted features can always be in high dimensionality. Therefore, performing BN can decrease the difference of the distribution of the features for objects from the same class, which increases the speed of training process. Additionally, BN forces the posterior neurons not to rely too much on the anterior ones, it can be regard as a method of regularization to lift the network’s generalization ability.

**5 Experiments**

**5.1 Data Preprocessing**

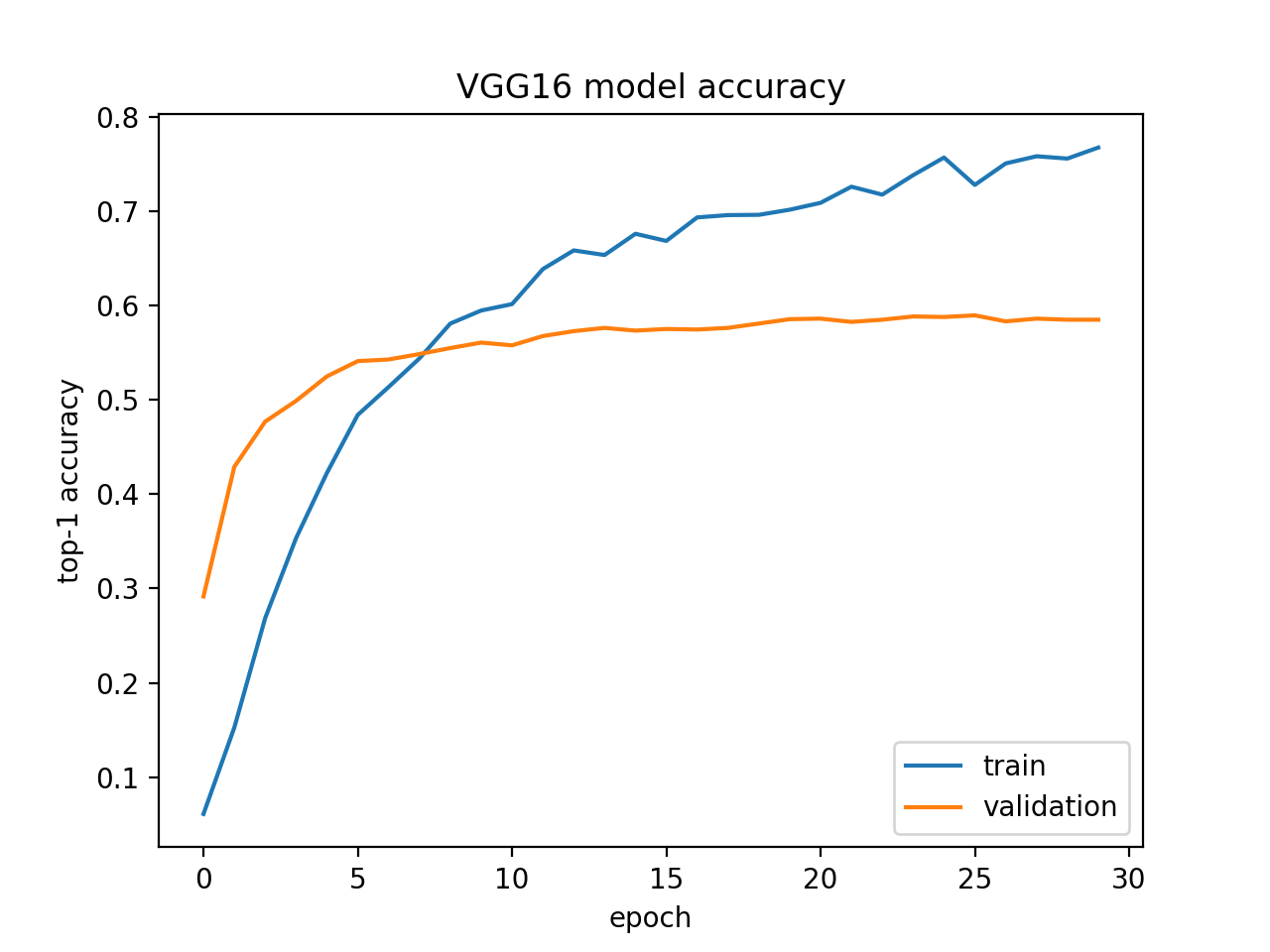
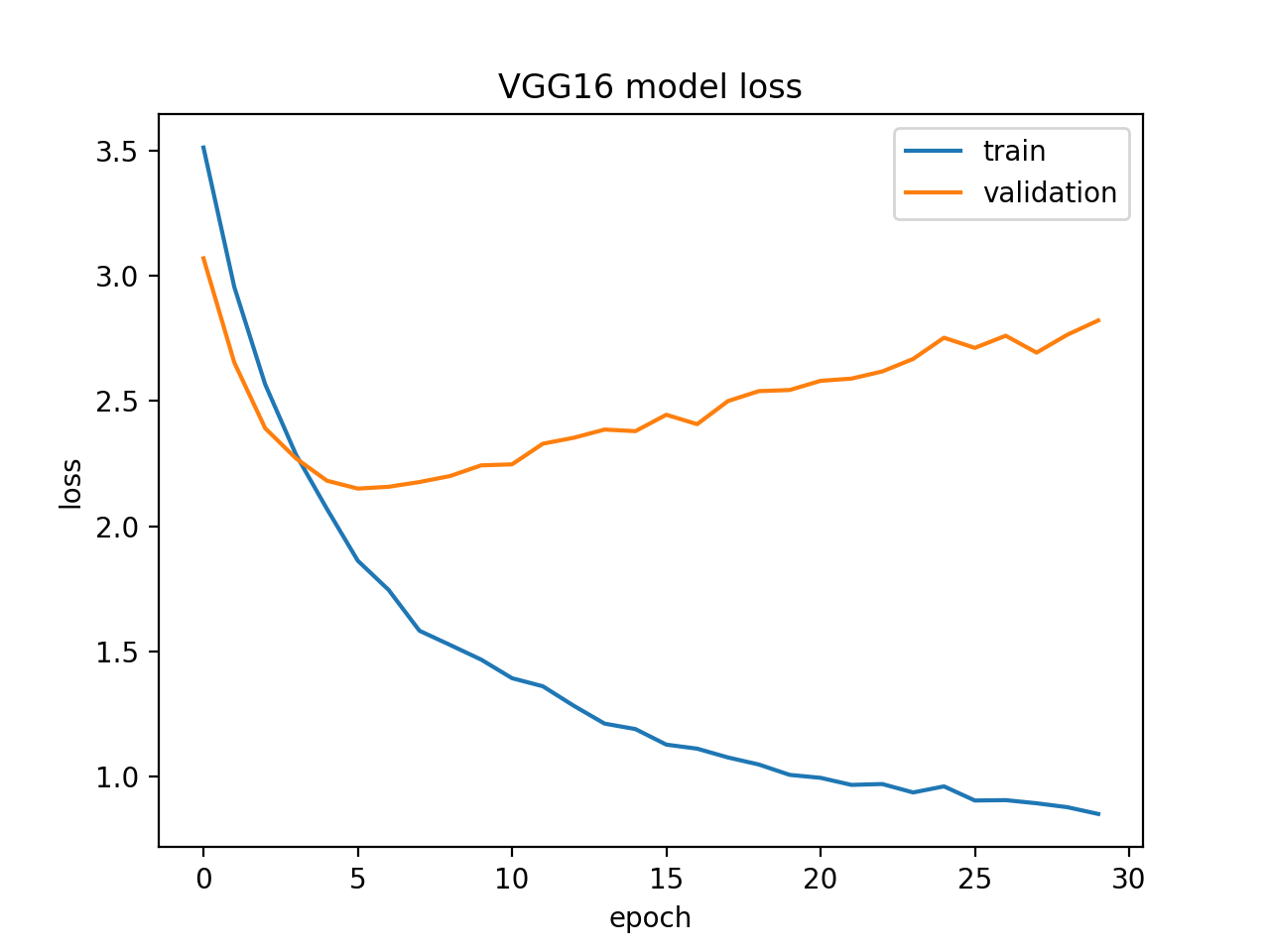
As we mentioned before, there are two different sets with 37 classes provided in this project, one contains the original pet images and the other contains the shape annotations of objects. Based on the ground truth value, we firstly split the files in both images sets randomly into train, validation and test folders with ratio 2: 1: 1 by coding. Then we manually divide images from train, validation and test folders into 37 classes. The reason why not use program to do it is that it costs so much time to create new sub-folders and move images into it. The following Fig.7 shows the result of separating the pets’ shape.



*Fig.7: Examples of segmented shapes.*

**5.2 Learning with VGG16**

In this project, we did not use a complete VGG16, which is because a complete VGG16 has 138 million parameters, which are too many for this project and will take too much time to get a good result. Due to some limitation on hardware, we decided to modify the VGG16 to have less parameters, like changing the fully connected layer’s units, which is a dimensionality of the output space, from 1000 into 37. Besides, reducing the some of the layers appropriately could also reduce the parameters of the VGG16, so, in this project, we use 12 layers instead of 16 layers. What is more, rather than using batch size equal to 32, we use batch size 20, which can also improve runtime a little because of taking up fewer memories. With doing these operations, the runtime of VGG16 could be reduced but at the same time, it could also bring some unpredictable results, like the overfitting. The following graphs are the accuracy and loss of using a modified VGG16 in our projects.

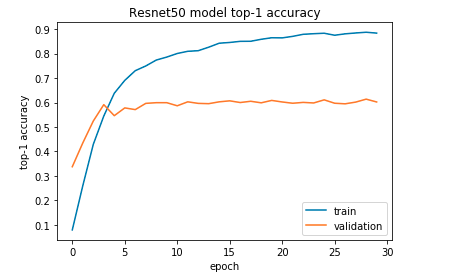
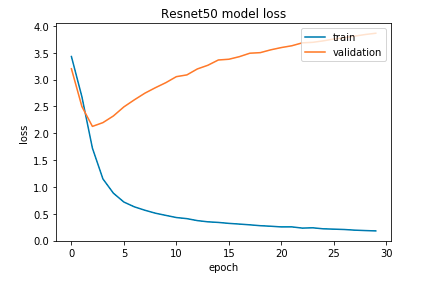


*Fig. 8: Learning curves in terms of top-1 accuracy (LEFT) and loss (RIGHT) of pre-trained VGG16 model during training and validation.*

**5.3 Learning with ResNet50**

Owing to the small dataset, we use the data generator to generate more data in order to avoid overfitting of the model and improve the accuracy of the result. Thus, the number of parameters of the model is increased to 7453278.

However, training deeper neural networks also requires more time. Hence, to implement the ResNet50 model, firstly we replace each 2-layer block in the 34-layer net with 3-layer bottleneck block. The reason why we are using bottleneck feature is to give us a more efficient model by saving the training time of the model. Additionally, by learning the feature in the images, it might as well improve the accuracy of the result. Following figures are the accuracy and loss of the model.



*Fig. 9: Learning curves in terms of top-1 accuracy (LEFT) and loss (RIGHT) of pre-trained ResNet50 model during training and validation.*

**5.4 Learning with Xception**

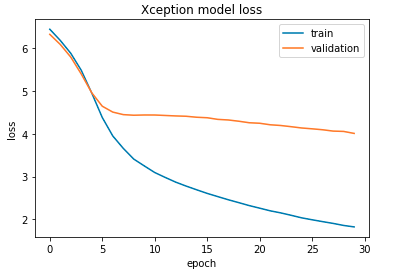
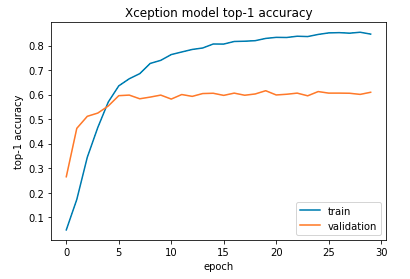
We apply transfer learning to build a CNN by using the pre-trained Xception model as the feature extractor. In the first part of the network, we only instantiate the convolutional part, and there is one node for each pet class in the fully-connected layer equipped with the softmax. Then we run this model on the training, validation and testing dataset separately once, and the last convolutional output features are recorded. In the second part, we calculate the predictions according to the extracted bottleneck features. In consideration of computational efficiency, we use the original labels to train a fully-connected model on the top of stored features. Finally, we classify all the features extracted from the pre-trained Xception models by using the same fully-connected classifier illustrated on the Fig. 10. Also, the batch size can significantly influence the accuracy. The smaller batch sizes make the model learn slower between epochs and reach higher maximum training accuracy than the larger batch size, and we determine the batch size as 4 at last.

A screenshot of a cell phone

Description automatically generated

*Fig. 10: The structure of the fully-connected classifier on the top of the pre-trained Xception.*

The learning curve graphs Fig.11 below show that the training accuracy can achieve over 80% and the validation accuracy can achieve at around 60% within 30 epochs, while the network starts overfitting after the 5th epoch.



*Fig. 11: Learning curves in terms of top-1 accuracy (LEFT) and loss (RIGHT) of pre-trained Xception model during training and validation.*

**6 Discussion**

**6.1 Choice of Models**

After tuning hyper-parameters and applying different methods for reducing the overfitting, we save the model and parameters that are learnt from the epoch with the lowest validation loss, which means the epoch is the last epoch before the model starts overfitting. Then we load the best model for each architecture to examine on the test set. The experiment results on the test dataset are summarized in Table 1 below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Top-1 Test  Accuracy | Top-5 Test  Accuracy | Parameters Count |
| CNN from scratch | 11.5673% | 35.0422% | 25,720,149 |
| VGG16 | 79.9876% | 96.2397% | 1,450,533 |
| ResNet50 | 89.0098% | 99.1320% | 7,453,278 |
| Xception | 89.9962% | 99.8435% | 1,404,581 |

*Table 1. Top-1 and Top-5 Test Accuracy for Different Pre-Trained Architectures.*

In the end, we choose the transfer learning based Xception model to implement the final pet breed classifier, because it gives out the highest top-1 test accuracy rate with the fewest parameters and the overfitting of this model seems to be the least serious. When given a cat or dog image that external of the dataset, we firstly use the pre-trained Xception model to predict it is dog or cat, which is convenient due to the fact that all dog categories are corresponding to the ImageNet1000 dictionary keys from 151 to 268[6]. Then we apply our best-performance model to predict the specific pet breed. Some examples are shown in the Fig. 12 below.

A cat that is looking at the camera

Description automatically generatedA dog running in the grass

Description automatically generatedA screen shot of a cat

Description automatically generatedA picture containing animal, mammal, cat

Description automatically generated

*Fig. 12: Classification results for images that are not in the original dataset.*

**6.2 Comparison of Methodology**

During the experiments, we explored three models without any overfitting reduction methodologies, yet generally all three models have quite serious overfitting issues. To boost the performance of the model and reduce the effects of overfitting, we conduct three methodologies as follows.

(1) *Dropout*: with a common choice of 0.5 dropout rate and other parameters left unchanged, the top-1 accuracy of model improves 0.45% and the top-5 accuracy of model improves 0.61%. This is not a huge improvement and it does not reach our expectation.

(2) *Batch Normalization*: with adding batch normalization into our model and does not change any other parameters, the top-1 accuracy of the model improves 0.32% and top-5 accuracy improves 0.47%. The result is similar to only use the Dropout methodology and it does not reach our expectation either.

(3) *Data Augmentation*: By using data augmentation in the model and unchanging any other parameters, the top-1 accuracy of the model increases 0.27% and the top-5 accuracy of the model increases 0.39%. The effects of this methodology are not as good as the other two methodologies.

**6.3 Overfitting**

Although it is pretty simple and possible to achieve high accuracy on training set, but what we really want is to construct a model that is generalized enough to the testing data. Overfitting occurs when we fit too closely to the data and thus affect the performance negatively as a result.

With doing all the methodologies, the problem of overfitting still exists for some reason. The problem can be shown on figures [8, 9, and 11], as we can see from the graphs, the loss of the result decreases dramatically from epoch 1-4 in all models. However, in VGG16 and ResNet50 models, the loss of the result increases afterwards while Xception model decreases continuously. As a result, the performance of Xception model is better than VGG16 and ResNet50. Moreover, it also shows that the models encounter the overfitting problem after the fourth epoch. Thus, to make sure that our models are not affected by the overfitting problem, we use the models that generated from 4th epoch to predict the result.

But even we use the pre-trained Xception model, it still has overfitting problem. Overfitting can occur when a model begins to memorize the training data instead of learning to generalize the data. Our model could encounter with this problem, although with the using of dropout, it can affiliate the problem, but problem might not be solved. It can still learn some specific feature instead of the most robust one, and that causes the overfitting of our result. On the other hand, if some features are shown in our train set but not in validation and test set, it can cause overfitting as well. For example, in our train set it includes attributes such as tail, ears, mouth, eyes, color, noses, teeth, and fur. It is easy to construct a model that is able to predict the breed given the other attributes. But what if in the test set it only contains the back of the cat or dog which only has the color and tail? And the model that we generated will not work this time, which causes overfitting as it is not generalized enough to the new data.

**7 Conclusion**

In this report, three different CNN models are explored and experimented in order to solve the pet breed identification challenge to classify images into 37 cat or dog categories. Dropout layers, batch normalization and data augmentation are used to reduce the problems of overfitting aiming to potentially further improve the performance of network. After experiments, the more lightweight and more advanced Xception architecture performs better when it comes to training accuracy and time. Fine-tuning Xception model gives out the top-1 test accuracy of 89.9% and the top-5 accuracy of 99.8%. For possible future works, apart from exploring other more stylish models, such as ResNet152 and Inception-V4, we would try to figure out the reasons for model’s overfitting, and hopefully eliminate it completely.

**8 Learning Outcomes**

In terms of what we have learnt from this project, on the one hand, we understand and apply several different models including VGG16, Resnet50 and Xception in CNN with three different over-fitting reduction methodologies in image classification. And based on the same data set, different models would have different effects. Besides, using different overfitting reduction methodologies to reduce the overfitting issue on one model could have different effects. On the other hand, the corporation among teammates are also quite important. Efficient corporation could increase the project quality and make sure the project to be completed in time, but ineffective corporation would primarily affect the process of the project and may delay the time schedule of the project.

**References:**

[1] Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1800-1807.

[2] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2818-2826.

[3] Xu, J. (2017). An Intuitive Guide to Deep Network Architectures. Retrieved from https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41

[4] Fine-grained image recognition - Convolutional features for visual recognition | Coursera. (2019). Retrieved from https://www.coursera.org/lecture/deep-learning-in-computer-vision/fine-grained-image-recognition-lBgu5

[5] Parkhi, O.M., Vedaldi, A., Zisserman, A., & Jawahar, C.V. (2012). Cats and dogs. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 3498-3505.

[6] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115, 211-252.

[7] Elson, J.E., Douceur, J.R., Howell, J., & Saul, J. (2007). Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. ACM Conference on Computer and Communications Security.

[8] Golle, P. (2008). Machine learning attacks against the Asirra CAPTCHA. IACR Cryptology ePrint Archive, 2008, 126.

[9] Liu, J., Kanazawa, A., Jacobs, D.W., & Belhumeur, P.N. (2012). Dog Breed Classification Using Part Localization. ECCV.

[10] Wang, X., Ly, V., Sorensen, S., & Kambhamettu, C. (2014). Dog breed classification via landmarks. 2014 IEEE International Conference on Image Processing (ICIP), 5237-5241.

[11] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ICML.

[12] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.R. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15, 1929-1958.

[13] Jay, P. (2018) Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook. Retrieved from https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cc5d0adf648e

[14] Jordan, J. (2018) Common Architectures in Convolutional Neural Networks. Retrieved from https://www.jeremyjordan.me/convnet-architectures/

[15] Raj, B. (2018) Data Augmentation | how to use deep learning when you have limited data. Retrieved from https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data- augmentation-c26971dc8ced

[16] Han, D. (2018) VGG16 notes, VGG16 structure image, Personal website. Retrieved from http://deanhan.com/2018/07/26/vgg16/

[17] Das, S. (2017) CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more. Retrieved from https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

**Confidential Peer Review**

|  |  |  |
| --- | --- | --- |
| **Group Member** | **Workload Distribution** | **Contribution Ratio** |
| **Yingjin Song (u6122877)** | Demo & Presentation  Implement the Xception  Research on Batch Normalization  Report Writing | **28%** |
| **Weitao Chen (u6309924)** | Demo & Presentation  Implement the VGG16  Research on Data Augmentation  Report Writing | **28%** |
| **Jinming Dong(u6683369)** | Demo & Presentation  Implement the Resnet50  Research on Dropout  Report Writing | **28%** |
| **Zirui Chen (u6227730)** | Demo & Presentation  Implement the Resnet | **16%** |