Final Project Report of Intruduction to Data Science – A Study of Famous Image Classification Models on Flower Classification Task

Weiyi Zhang
1600017749
Yuanpei College, Peking University

Qianli Shen
1500017740
Yuanpei College, Peking University

1 Team Info

Our team name is '12138', a magic number we chose as a random seed for validation.

Leader Zhang(1600017749) implemented Naive CNN part and Transfer Learning part.

Member Shen(1500017740) implemented the rest.

Thanks to Deep Learning Lab, Big Data Scientific Research Center of Peking University for computational resource supports. Thanks to our classmate Haodong Duan(Kenny) for advices.

2 Task

In this flower classification task, we need to train our model based on a total of 3,899 images of five species of flowers, and give the type of 424 flowers in the test set to get as much accuracy as possible.

The data is from `https://www.kaggle.com/alxmamaev/flowers-recognition`

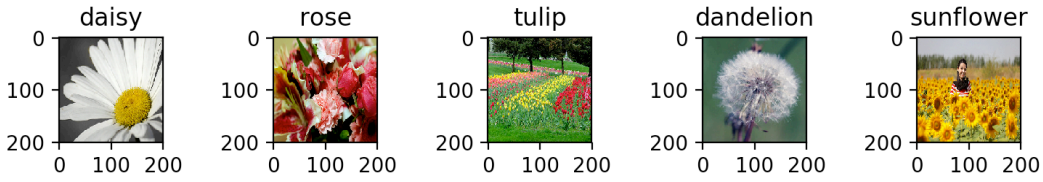


Figure 1: data

The task is announced as a kaggle competition. As a part of kaggle competition rules, labels of test set are invisible and the public leaderboard result is calculated with approximately 50% of the test data. The final rank is calculated with the rest 50%.

3 Model

3.1 Convolution Neural Networks (CNN)

CNN is a class of deep neural networks, which is commonly used in visual imagery. It typically includes convolution layers, pooling layers and fully connected layers. Each neuron receives input from the previous layer, and to restrict numbers of parameters to speed up the training process and control overfitting, it adopts two strategies:

- **Sparse Local Connectivity**. CNNs enforce a local connectivity pattern between neurons of adjacent layers, and thus the layer creates representations of small inputs given by the previous layer.

- **Shared Weights**. To further reduce the number of parameters, CNNs replicate each filter across the entire visual field. Each units share the same weight vector and bias, and form a feature map.

3.2 Deep CNN (VGG)

VGG16 uses very small (3×3) convolution filters and increases the depth to 16 layers in total, where 13 convolutional layers are stacked one after the other and 3 dense layers for classification.

- The filters for convolution layers are grouped into five, connected by max pooling layers.

- The number of the filters in each group are increasing, taken from the sequence of [64, 128, 256, 512, 512]. The dense layers comprise of 4096, 4096, and 1000 nodes each.

VGG19 is similar in architecture with one additional convolution layer for each group in the last three. The use of (3×3) convolutions with stride 1 gives an effective receptive filed equivalent to (7×7), which implies there are fewer parameters to train.

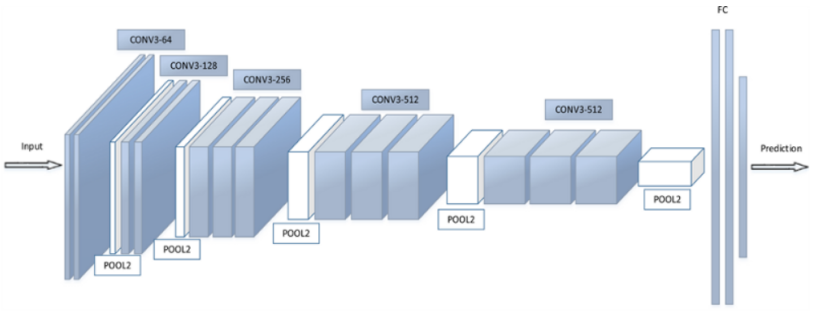


Figure 2: VGG16

3.3 Inception

The key idea of inception networks is to design good local network to between the two layers, which is called **Inception modules**.

- Inception modules act on the same layer, using multi-level feature extractor with convolutions of different sizes.

- To perform dimension reduction, it adds 1×1 convolutions before 3×3, 5×5 convolutions and after 3×3 max pooling.

- It adds two auxiliary classification outputs during the generation of the network, which are used to inject gradients at lower layers.

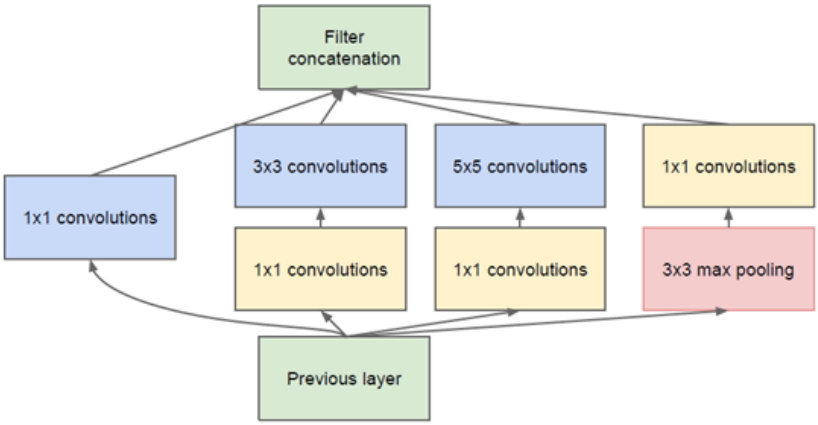
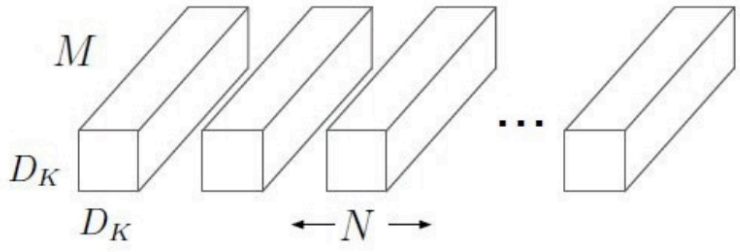


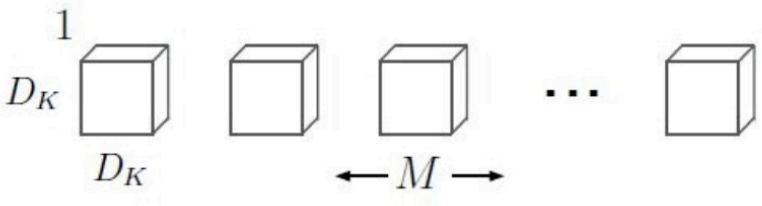
Figure 3: Inception

3.4 Xception

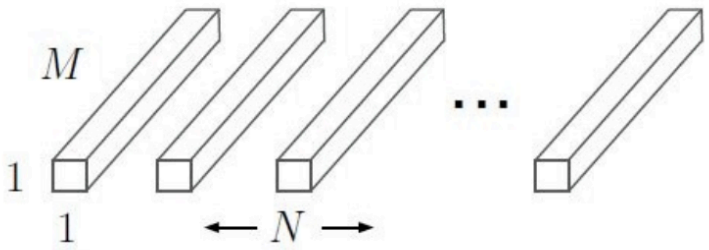
Xception is an extension of the Inception network, while it replaces the Inception modules with depthwise separable convolutions, which reduce the number of parameters further and allow for deeper network.
Suppose we have M feature maps and we want to get N features maps in the next layer. Traditional way would be to use N 3 filters, while depthwise separable convolution uses M 3 filters to generate M intermediate feature maps and then use N 1×1 Pointwise Convolution to generate the desired N feature maps.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

3.5 ResNet

All the previous models used deep neural networks in which they stacked many convolution layers one after the other. It was learnt that deeper networks are performing better. However, it turned out that this is not really true. Following are the problems with deeper networks:

Network becomes difficult to optimize Vanishing / Exploding Gradeints Degradation Problem (accuracy first saturates and then degrades) Skip Connections So to address these problems, authors of the resnet architecture came up with the idea of skip connections with the hypothesis that the deeper layers should be able to learn something as equal as shallower layers. A possible solution is copying the activations from shallower layers and setting additional layers to identity mapping. These connections are enabled by skip connections which are shown in figure 4.

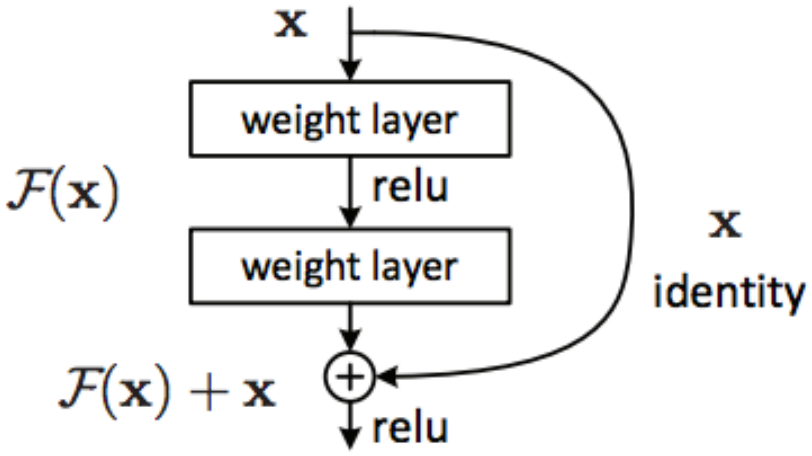


Figure 4: resdual block

So the role of these connections is to perform identity function over the activation of shallower layer, which in-turn produces the same activation. This output is then added with the activation of the next layer. To enable these connections or essentially enable this addition operation, one need to ensure the same dimentions of convolutions through out the network, that's why resnets have same 3 by 3 convolutions throughout.

By using residual blocks in the network, one can construct networks of any depth with the hypothesis that new layers are actually helping to learn new underlying patterns in the input data. The authors of the paper were able to create the deep neural network architecture with 152 layers. The variants of Resnets such as resnet34, resnet50, resnet101 have produced the solutions with very high accuracy in Imagenet competitions.

3.6 DenseNet

The main idea is to connect each layer to every other layer in a feed-forward fashion.
While traditional convolution networks with L layers have L connections, one between each layer and its subsequent layer, DenseNet has $\frac{L(L+1)}{2}$ direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. The advantages of DenseNet include:

- Alleviate the vanishing-gradient problem.
- Strengthen feature propagation.
- Encourage feature reuse.
- Substantially reduce the number of parameters.

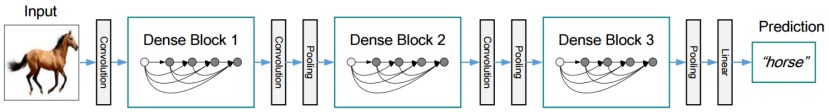


Figure 5: DenseNet

3.7 InceptionResNet

Learning the model architectures directly on the dataset of interest on a large data set can be expensive, thus the work proposes to search for an architectural building block on a small dataset and then transfer the block to a larger dataset and design a new search space which enables transferability.
Meanwhile, it introduces a new regularization technique called ScheuledDropPath to improve its generalization.

3.8 Architecture Search and NASNet

Learning the model architectures directly on the dataset of interest on a large data set can be expensive, thus the work proposes to search for an architectural building block on a small dataset and then transfer the block to a larger dataset and design a new search space which enables transferability.
Meanwhile, it introduces a new regularization technique called ScheuledDropPath to improve its generalization.

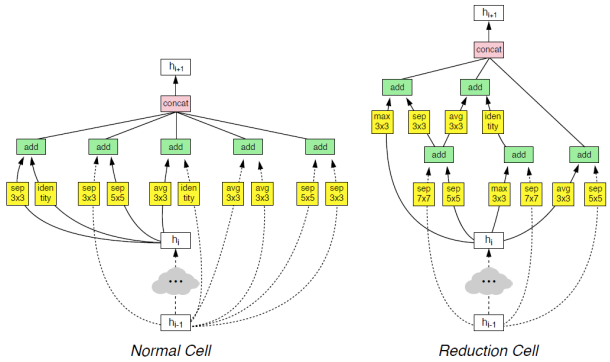


Figure 6: NASNet

4 Experiments

This flower classification task is small-scale compared with other famous image classification tasks like Imagenet and Cifar. Overfitting will occur in deep and confused models due to the small size of

train set. So we first try some simple models like Naive CNN. Simple models are computationally enconomical, easy-training and time-saving, but powerless.

Fortunately, researchers who raised models above provide open source implementations and Imagenet pre-trained weights of their models. So we can use pre-trained models to do transfer learning and finetuning.

Finally we ensemble results of multiple models up to get our submissions to kaggle competition.

4.1 Naive CNN

Simple shallow convolution nueral networks behave well in small-scale tasks like MNIST. So we designed a 6-layer convolution nueral networks to see how simple models behaves in this task.

4.2 Feature Extraction & Transfer Learning

Models pretrained on large-scale classification tasks have learned to extract important features to do classification. One idea to take advantage of pretrained models is deriving the output of some layers as features and train new models to fit the labels.

We removed the top fc layer of ResNet50 and derived 2,048-dimension features of each image. Then we designed a 2-layer fully connected nueral network to do classification. Although we used complex models to extract features but the indeed trainable classifier can be simple. The performance is markedly better than Naive CNN, but far from finetuning models, which will be detailed in next subsection.

4.3 Finetune

If we consider the transfer learning model from another sight, we can regard it as a complete model with only top trainable. Now we unfreeze whole parameters and train the model on training data by low learning rate. In this way we can make full use of pretrained models and fit it better on data.

We slightly modefied open source codes to build our finetune program. Models we have tried include vgg16, vgg19, InceptionV4, Xception, Resnet50, Resnet101, Resnet151, Densenet121, Densenet169, Densenet161, InceptionResnetV2. We selected promising ones according to validation accuracy rate for prediction.

4.4 Data Augmentation

Models may overfitting in train set for lack of training data. Adding extra data is not permitted, but we can apply ZCA whitening, randomly rotate, zoom, shift or flip images, which we call data augmentation, to help us ease overfitting with limited train data.

Keras provides practical data generator to generate augmented data in batches. Here we show some augmented data as examples.

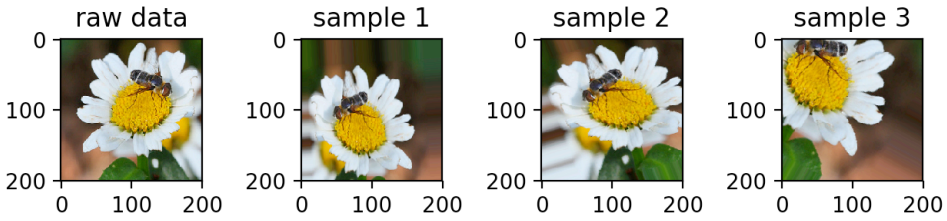


Figure 7: augment samples

4.5 Model Selection

We choose cross validation as our model selection method. Every time we randomly split training data by 20% as validation set to see how models fit in the rest part of data. Except model architecture,

we also tried different fc-layers(size, pooling method, dropout rate), normalization methods, data generators and optimizers.

4.6 Ensemble

Models make some mistakes in prediction, some of which are caused by variance. Let's consider three Resnet50 models and tree test cases. Model A gives correct prodiction in case 1 and 2, but makes mistake in case 3. Model B is wrong only in case 1 and C in case 2. Each single model achieve 66.67% validation accuracy. However if we ensemble three predictions up we will get all correct prediction. So we train multiple models of same architecture and ensemble the outputs by average weight to avoid the influence of validation.

We can also ensemble the outputs from models of different architectures. We guess different models may 'see' different things from a same image. We are not sure about it but in this way we got the best predicion in public board.

4.7 Result

model	size/MB	param/1e6	val acc	test acc	ensemble acc
Naive CNN	39.2	9.8	0.822	0.811	-
Resnet50(TL)	0.04	0.01	0.869	0.849	-
vgg16(FT)	528	138.4	0.883	-	-
vgg19(FT)	549	143.7	0.901	-	-
Xception(FT)	92	22.3	0.963	0.948	0.958
Resnet50(FT)	103	25.6	0.949	0.939	0.950
Resnet101(FT)	179	44.8	0.937	-	-
Resnet151(FT)	243	60.8	0.951	0.955	0.955
Desnet121(k=32)(FT)	34	8.1	0.956	0.948	0.948
Desnet169(k=32)(FT)	59	14.3	0.951	-	-
Desnet161(k=48)(FT)	118	28.7	0.960	0.958	0.960
InceptionResnet(FT)	225	56.3	0.965	0.946	0.953
Final Submission	-	-	-	-	**0.972**

Some blocks remain blank because these models seem to be weak according to validation accuracy, or because there are congeneric models with better performance so we gave them up for further experiments.

Our final submission combines results from multiple models and achieve highest accuracy of all.

References

[1] Simonyan, K. , & Zisserman, A. (2014) Very deep convolutional networks for large-scale image recognition. Computer Science.

[2] Chollet, François. (2016) Xception: deep learning with depthwise separable convolutions.

[3] He, K. , Zhang, X. , Ren, S. , & Sun, J. . (2015). Deep residual learning for image recognition.

[4] Huang, G. , Liu, Z. , Laurens, V. D. M. , & Weinberger, K. Q. . (2016). Densely connected convolutional networks.

[5] Zoph, B. , Vasudevan, V. , Shlens, J. , & Le, Q. V. . (2017). Learning transferable architectures for scalable image recognition.

[6] C Szegedy S Ioffe V Vanhoucke (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning