Classifying Images as Dogs or Cats with Deep Learning

- ML Final Project

Yue Zhang

GitHub Link of this project: https://github.com/EllieZhangy/Classify-Images-as-Dogs-or-Cats

Agenda

Problem Statement

01

Background:
For the recognition of dogs and cats images, due to the great diversity of the photo database (backgrounds, angles, poses, lights, etc.), it is difficult to perform accurate automatic classification.

It has been shown in the literature that machine classifiers can achieve an accuracy of more than 80% on this task [1].

02

Problem Statement:
In this project, **deep learning methods** are used to **classify whether images contain either a dog or a cat.**

Project Summary: The project involves **a combination of custom CNN architecture , Transfer Learning techniques (InceptionNet as the pretrained models) and model improvement methods,** to develop a deep learning solution for the **dogs and cats image classification task**.

Data and Methodology

Data

Data Source:

854.51MB Image Data from Kaggle
https://www.kaggle.com/competitions/dogs-vs-cats-redux-kernels-edition/data

Data Description:

- **Train folder:** 25,000 images of dogs and cats. Each image in this folder has the label as part of the filename.
- **Test folder:** 12,500 images (without label, need to be predicted), named according to a numeric id.

Methodology

1. Custom CNN from Scratch

- **Custom CNN Architecture**

2. Transfer Learning

Feature Extraction on Pretrained Model:

- **InceptionNet**

3. Model Improvement

- **Fine-tuning**
- **Regularization**
- **Hyper-parameter Tuning**

Exploratory Data Analysis

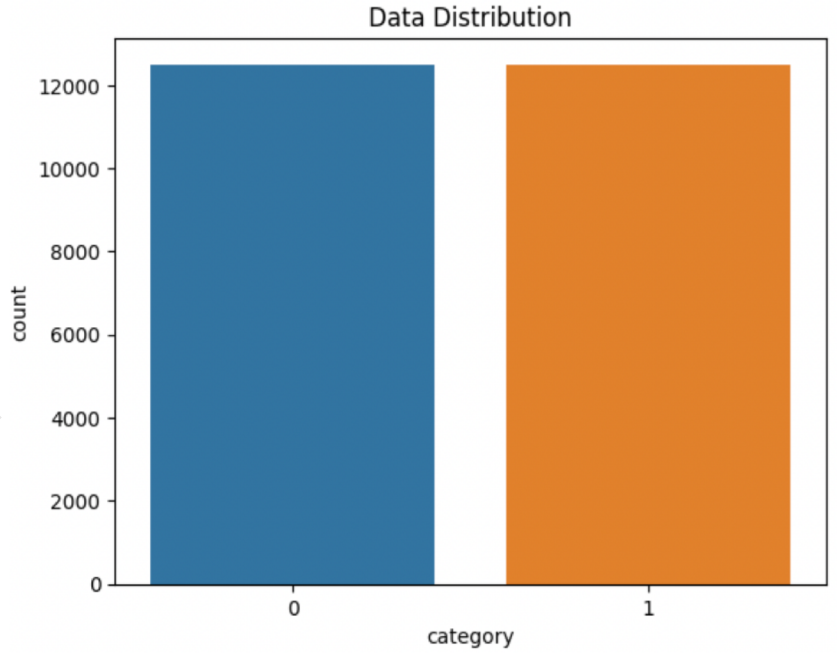
01 Check NULLs:

There is no missing values

02 Data Distribution

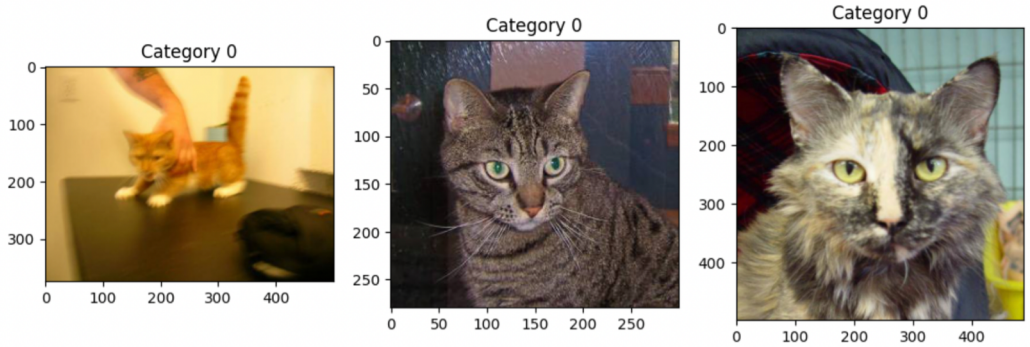
Formate: Generate the respective labels for dogs(1) and cats(0) for the training data

12500 dogs images and 12500 cats images (balanced)

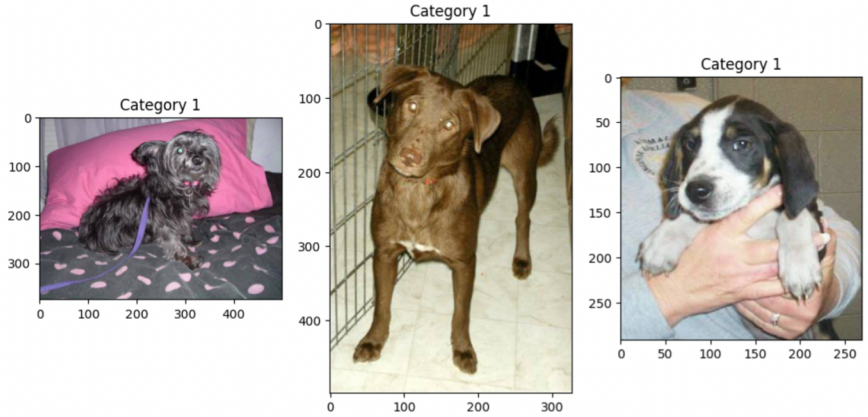


03 Some Input Samples:

Cats:



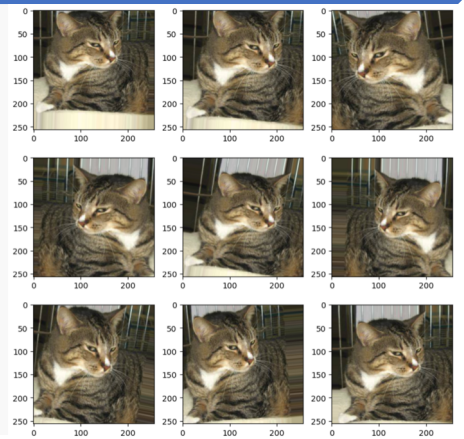
Dogs:



Data Augmentation and Feature Engineering

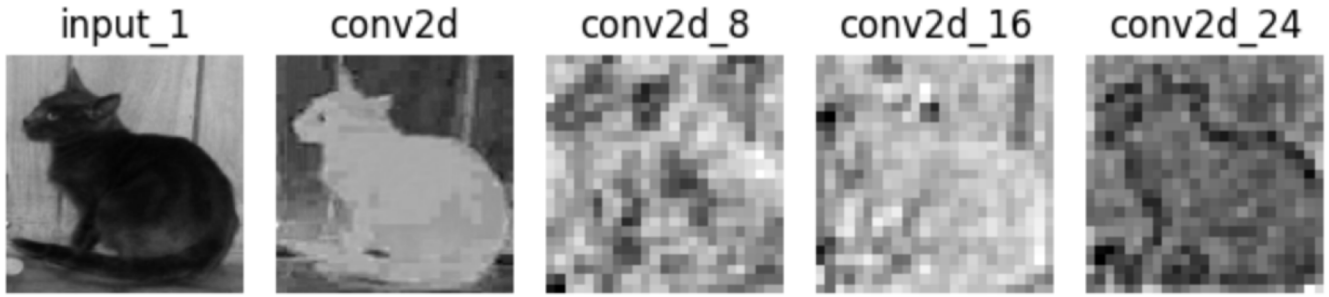
Data Augmentation :

- **Resize the images to a consistent shape**
- **Split the training data into training and validation set**
- **Generate mini batches** for training and validation set accordingly
- **For the training set, wrap the data generated with transformation for data augmentation.**



Feature Engineering (use InceptionNet as an example) :

Visualize the feature maps at different layers of InceptionNet based model for a given image:



Custom CNN from Scratch

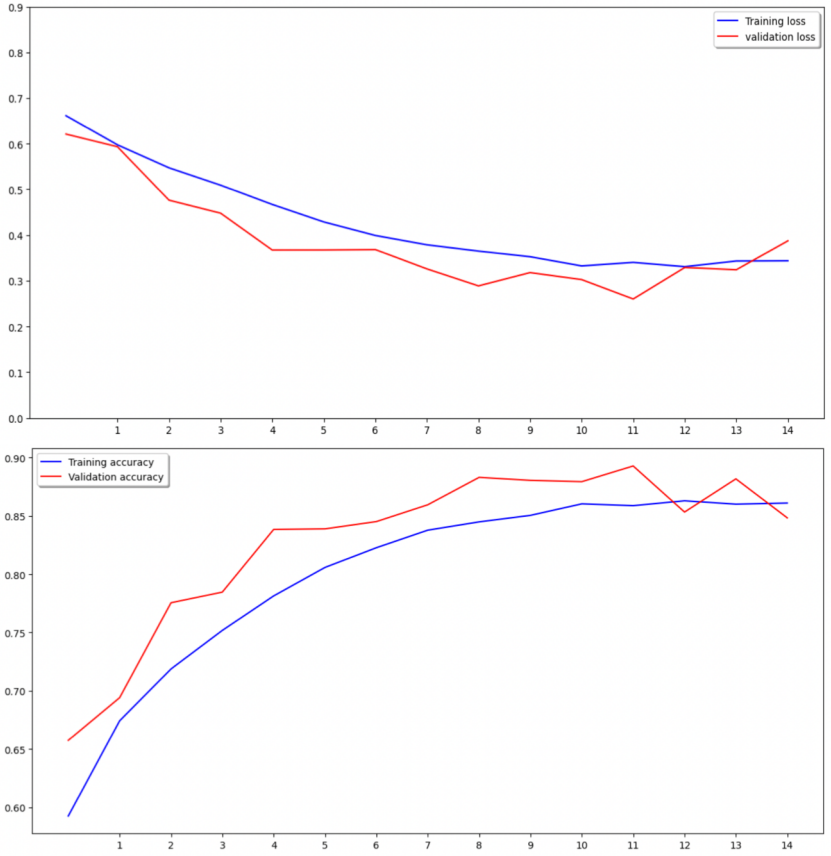
Custom CNN Architecture :

- **3 convolutional layers** with all having the same filter size of (3x3)
- **Max pooling layer**: Each convolutional layers is followed by a Max pooling layer that will reduce the dimensions of the convolutional features maps
- **Fully connected layer:** The output of the final convolutional layer will be passed into a Fully connected layer with 500 neurons
- **Last layer which acts as sigmoid activation** to predict 1/0
- **Compilation**

30 epochs
batch size: 16
(Early stopping at Epoch 15)

On validation set:
- **Accuracy: 0.848237**
- **Loss: 0. 387396**

On training set: (Accuracy/Loss vs. Epoch)



Transfer Learning

Custom CNN from scratch	VS	Transfer Learning
Train the custom CNN from scratch on the task without utilizing any pre-trained weights.		Involves leveraging pre-trained models that have been trained.

Transfer Learning has advantages in training time, data requirements, performance and generalization

Transfer Learning ： (use Feature Extraction in this project)

Step1

Image Feature Extraction: Use pretrained model as a fix feature extractor to predict output features

Step2

Training a Classifier: Feed the output features into a separate classifier algorithm to generate prediction

(Pretrained models used in this project**: InceptionNet)**

Transfer Learning – InceptionNet

Architectures of Transfer Learning Model Pre-trained on InceptionNet:

InceptionNet Based Model	
Base Model：	**Pre-trained InceptionV3**
Custom Layers on Top of the Base Model:	• Global Average Pooling2D layer (reducing the spatial dimensions to a single vector) • Dense layer with 1024 neurons and ReLU activation function • Dense layer with single neuron and sigmoid activation function
Freezing Base Model Layers (preventing models from being updated during training)	
Compilation	

Transfer Learning – InceptionNet

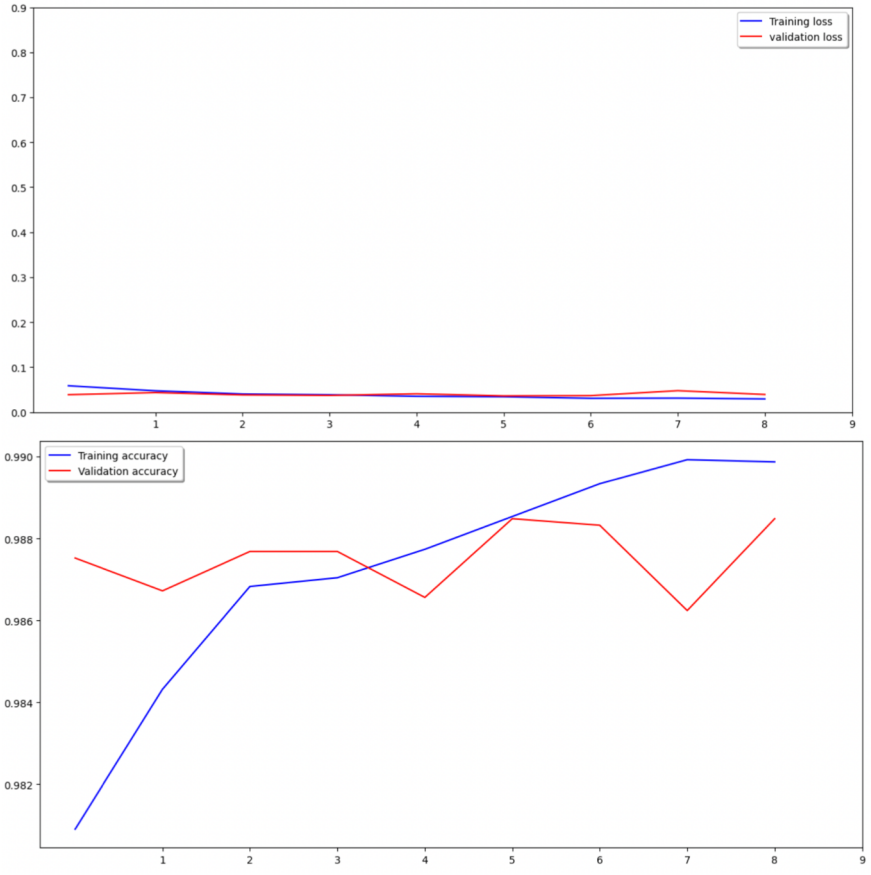
Performance of InceptionNet-based Model:

30 epochs
batch size: 16
(Early stopping at Epoch 9)

On validation set:
- **Accuracy: 0.988462**
- **Loss: 0. 039070**

- **From the two plots, the model's performance seems promising, with high and stable accuracy and low loss for both training set and validation set.** (Note the range of the y-axis in the second figure, the lowest accuracy at the first epoch has already reached 98%)
- **This model may have an overfitting problem,** because as the epoch increases, the training accuracy gradually exceeds the validation accuracy, but fortunately the gap between the two is not large.

On training set: (Accuracy/Loss vs. Epoch)



Model Comparison and Evaluation

Model	Validation Accuracy	Validation Loss
Custom CNN from Scratch	0.848237	0.387396
InceptionNet -Based	0.988462	0.039070

- **The transfer learning achieved significantly higher accuracies compared to the custom CNN models from scratch**, which demonstrates **the effectiveness of leveraging pre-trained models** for the tasks.
- Compared with Custom CNN model, **InceptionNet achieved the higher accuracy and lower loss on the validation set.**

Choose InceptionNet as the final model and conduct further model improvement

Model Improvement

Based on Inception Model:

- All the improvements methods and effects are listed below.
- As it is discussed in the previous slide that **there may exist an overfitting problem** for this model, so apart from tuning, **regularization is also necessary in this case**.

1. Fine-Tuning:

- Unfreeze the last 10 layers of the base model

2. Regularization :

- Add regularization techniques including Dropout and Batch Normalization (between the Global Average Pooling2D layer and dense layer).

3. Hyper-parameter tuning :

- Find the optimal setting for the model

After

- **Validation Accuracy: 0.985417**
- **Validation Loss: 0. 040710**
- **Lower gap between the performance on train and validation set**

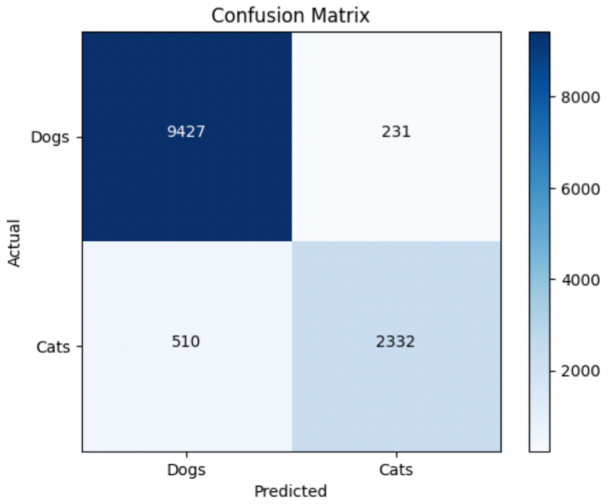
- **Best learning rate: 1e-05**
- **Best batch_size: 80**
- **Best accuracy on validation set: 0.99088**

Results

Analyze the performance and predicting results of the final model after tuning:

Performance on the Validation Data

- **One-time Validation Accuracy: 0.94072**
- **Best accuracy on validation set after hyperparameter tuning: 0.99088**
- **Confusion Matrix:**



Predict on the Test Data

Sine the test set is different from validation set, with no label known. Therefore, here I **visualize the result of predicting 9 images** randomly selected from the test data:

The Inception-Based Model makes pretty good prediction on all the 9 images.

Future Work

- **Further Analysis of Model Performance:** For example, InceptionNet has reached to a very high accuracy on both the training and test data and there is no significant gap after regularization. But there may still exist overfitting concern and in this case, cross-validation and error analysis could be conducted to further understand the model performance.
- **Exploring Advanced Architectures**: While we have achieved impressive results with models like InceptionNet, there are several other advanced architectures worth exploring such as DenseNet, EfficientNet, and Xception which always show promising results in image classification tasks.
- **Adaptive Learning Rate Scheduling**: While hyperparameter tuning helped us find optimal values for learning rate, we can further investigate adaptive learning rate scheduling technique, such as Cyclical Learning Rates, Learning Rate Decay, or One-Cycle Policy that can dynamically adjust the learning rate during training to improve convergence.
- **Ensemble Methods and Model Stacking**: We could also build an ensemble of our models, e.g. combining predictions from difference Inception models or different architectures.
- **Interpretability:** Investigating interpretability techniques like Grad-CAM, or Attention mechanisms to gain insights into which parts of the image contribute to the model's predictions.

Reference

[1] Golle P . Machine learning attacks against the Asirra CAPTCHA[C]// Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008. ACM, 2008.
[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
[3] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
[4] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
[5] Kaggle competition: Dogs vs. Cats Redux: Kernels Edition. https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition
[6] https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

GitHub Link of this project: https://github.com/EllieZhangy/Classify-Images-as-Dogs-or-Cats