CS5242 Dog Breed Classification

Liu lu, Kaiyu, Liuhuo, Jiang Liu

April 16, 2019

Outline

Background

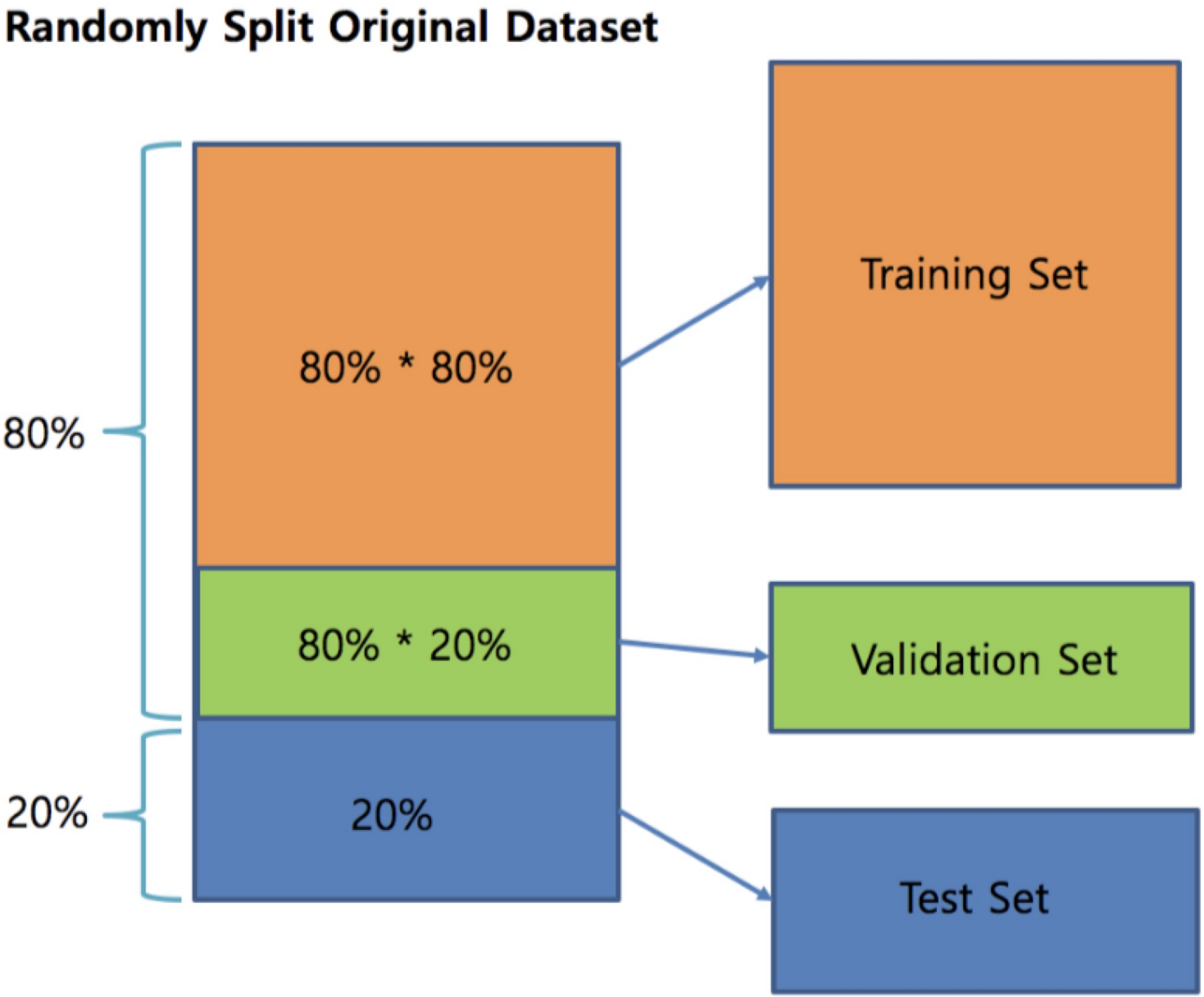
Homeless dog

Approximately millions of stray dogs per year worldwide, to help them find a home and improve the work efficiency for dog adoption center staff by updating the manual process of dog classification to an automated process

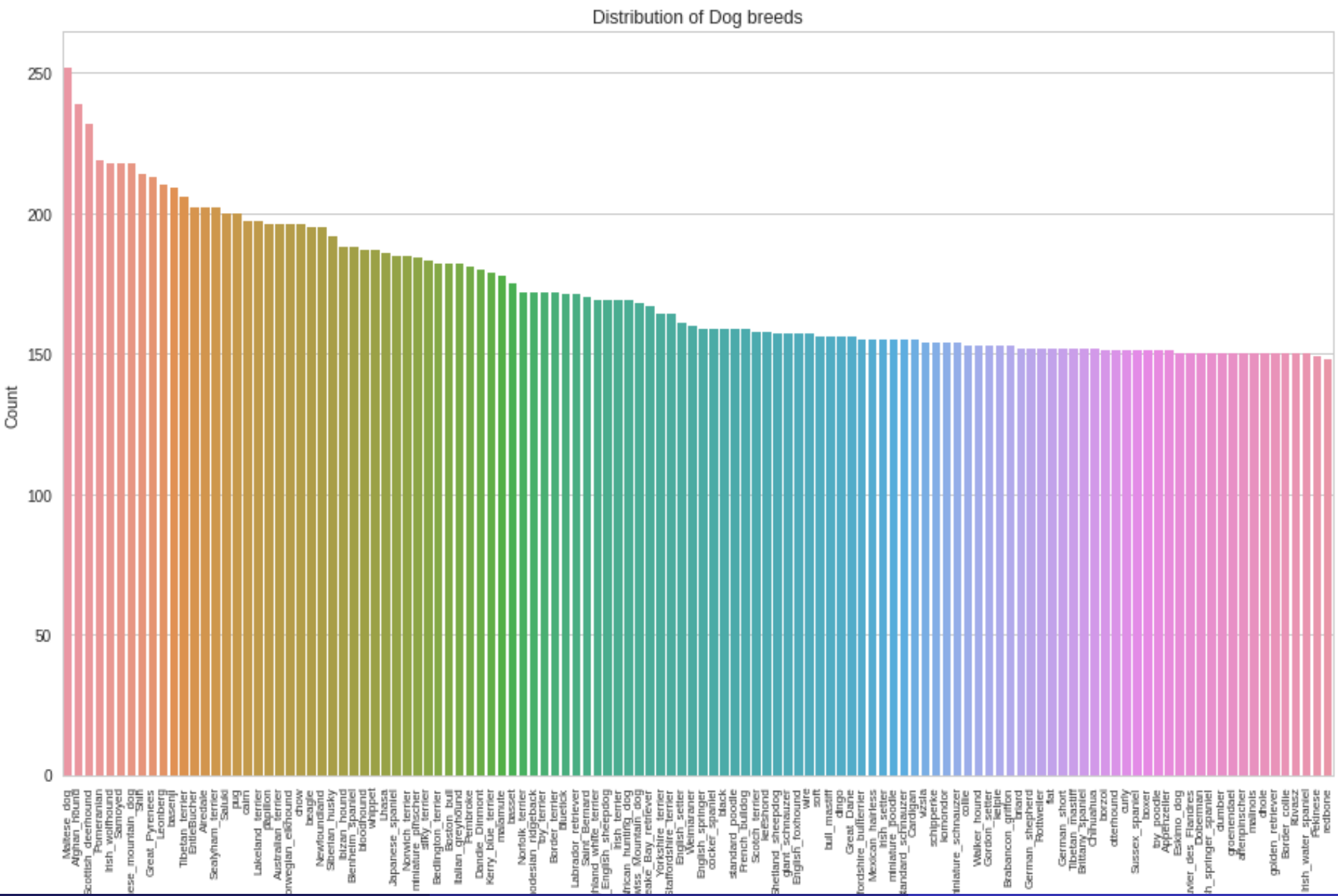
Dataset Analysis

- approximate 20000 pictures
- 120 class of dog breed
  `http://vision.stanford.edu/aditya86/ImageNetDogs/`
- about 200 pictures of each class

Data Preparation



Data Feature Visualization



Data Augmentation

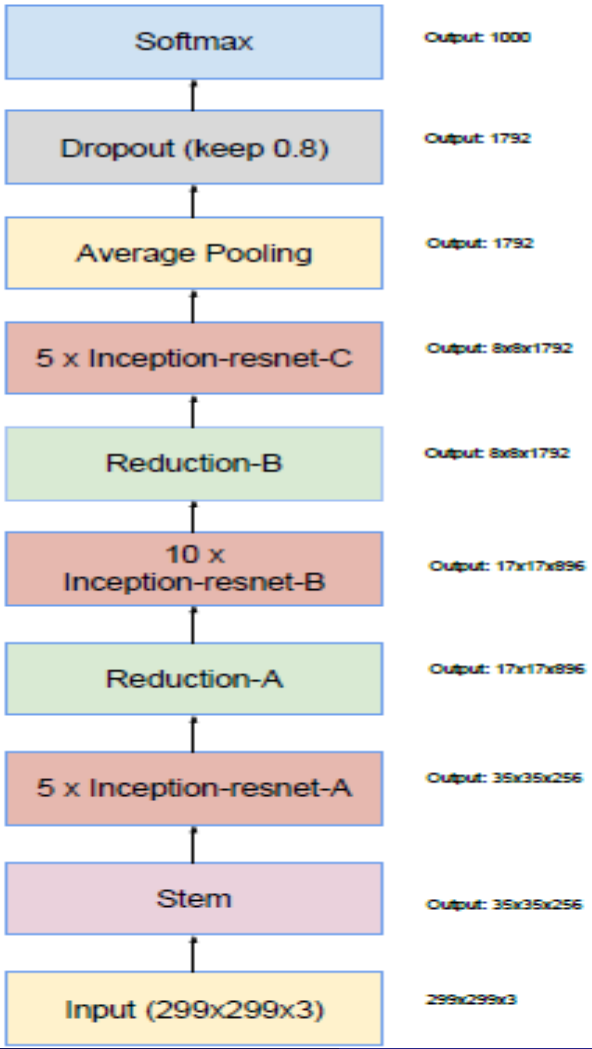
- rotation
- shear
- channel shift
- width shift
- height shift
- zoom range
- horizontal flip
- vertical flip

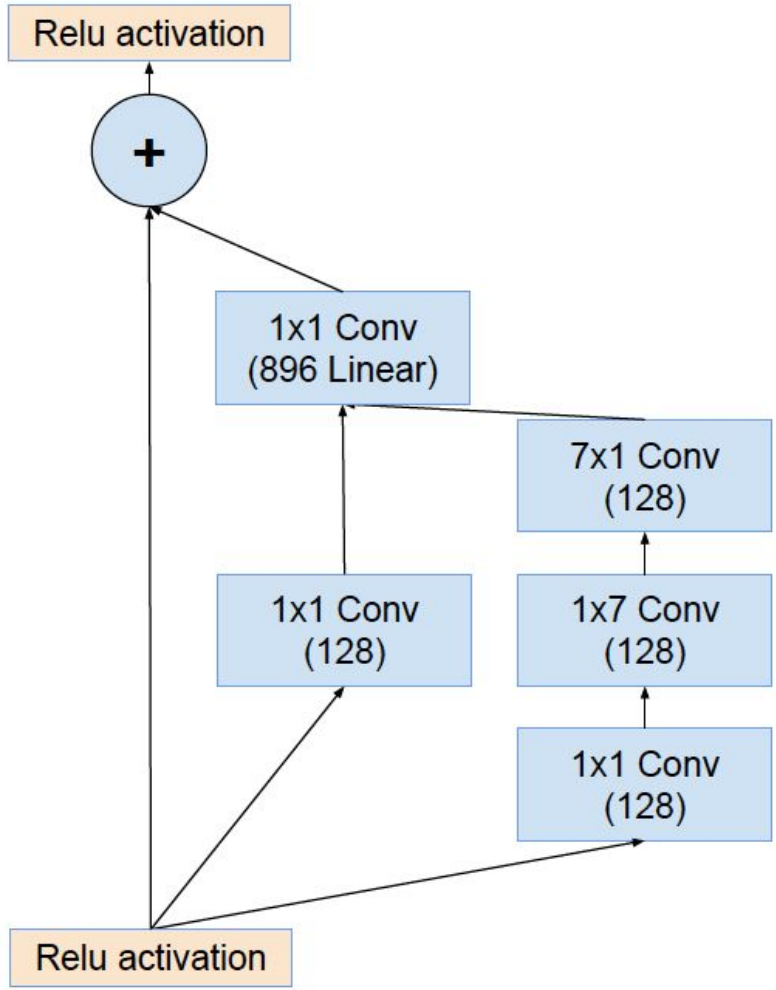
Model Comparison

Mode	training accuracy	val accuracy	test accuracy
VGG16	0.2401	0.2247	0.2119
ResNet50	0.9867	0.5272	0.5344
Xception	0.9985	0.6486	0.6568
InceptionV3	0.9744	0.5833	0.6011
InceptionResNetV2	0.9951	0.7231	0.7261

InceptionResnetV2 Model

- Add residual connection to Inception
- Compared to V1, V2 is larger and more precise





Transfer learning

Last layer:

- GlobalAveragePooling2D()(x)
- Dense(number)(x)
- BatchNormalization()(x)
- Activation('relu')(x)
- Dropout(0.5)(x)
- Dense(numberClasses, activation='softmax')(x)

Fine tuning

Version 1

```
model.add(GlobalAveragePooling2D())
model.add(layers.Dense(1024, activation = "relu"))
model.add(layers.Dense(120, activation = "softmax"))
```

accuracy (input size = 200): train: 0.992 val: 0.7346 test: 0.7430

Version 2

```
model.add(GlobalAveragePooling2D())
model.add(layers.Dense(1024))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(layers.Dense(120, activation = "softmax"))
```

accuracy (input size = 200): train: 0.99976 val: 0.66323 test: 0.6640
accuracy (input size = 299): train: 0.9983 val: 0.8224 test: 0.8207

Fine tuning

Version 3

```
model.add(BatchNormalization())
model.add(Dropout(0.001))
model.add(GlobalMaxPooling2D())
model.add(layers.Dense(512, activation = "relu"))
model.add(BatchNormalization())
model.add(Dropout(0.001))
model.add(layers.Dense(120, activation = "softmax"))
```

accuracy (input size = 200): train: 0.9951 val: 0.7231 test: 0.7262
accuracy (input size = 299): train: 0.9981 val: 0.8221 test: 0.8141

Version 4

```
model.add(Flatten())
model.add(Dropout(0.001))
model.add(layers.Dense(120, activation = "relu"))
model.add(BatchNormalization())
model.add(Dropout(0.001))
model.add(layers.Dense(120, activation = "relu"))
model.add(BatchNormalization())
model.add(Dropout(0.001))
model.add(layers.Dense(120, activation = "softmax"))
```

accuracy (input size = 200): train: 0.9542 val: 0.7668 test: 0.7774
accuracy (input size = 299): train: 0.9983 val: 0.7647 test: 0.7602

Result

```
Epoch 18/20
412/412 [==============================] - 323s 783ms/step - loss: 0.0057 - acc: 0.9986 - val_loss: 0.8109 - val_acc: 0.8154

Epoch 00018: ReduceLROnPlateau reducing learning rate to 1.56249996052793e-06.

Epoch 00018: val_acc did not improve from 0.82235
Epoch 19/20
412/412 [==============================] - 321s 780ms/step - loss: 0.0043 - acc: 0.9988 - val_loss: 0.7665 - val_acc: 0.8187

Epoch 00019: val_acc did not improve from 0.82235
Epoch 20/20
412/412 [==============================] - 322s 781ms/step - loss: 0.0056 - acc: 0.9985 - val_loss: 0.7880 - val_acc: 0.8120

Epoch 00020: ReduceLROnPlateau reducing learning rate to 7.81249980263965e-07.

Epoch 00020: val_acc did not improve from 0.82235
```

+ Code + Markdown

```
# accuracy metric 1
from keras.metrics import categorical_accuracy
result = model.evaluate_generator(test_gen,steps = test_steps)
print(model.metrics_names)
print(result)
```

```
['loss', 'acc']
[0.7806696782161937, 0.8206997085127353]
```

Conclusion

Challenges

- Unforeseeable hyper-parameter tuning result
- Evaluation difficulty due to long duration to train model

Future works to improve model accuracy

- Add boosting
- Combine different models
- Tune hyperparameter better

Thank you

Question?