City University of Hong Kong

2024 – 2025 Sem B

EE4146

Project Report

Transfer Learning Study

(Cat vs Dog Classification)

Name : Chu Lok Cheong *

Abstract

In this project, I used transfer learning of MobileNetV2 to develop a cat and dog image classification system. The model's dataset contains 10,000 images, and its validation accuracy reaches 96.5%. In this project, the effectiveness of transfer learning for small and medium-sized datasets was demonstrated through pre-trained weights and fine-tuning. The system is lightweight, fast, and suitable for deployment on edge devices.

Background

In this project, we developed a deep learning model to classify images as either cats or dogs. Using transfer learning of MobileNetV2, the model is trained using a dataset of cat and dog images and enhanced to improve generalization capabilities. The project includes using Grad-CAM for data preprocessing, model training, evaluation and interpretability, and saving the final model for future predictions.
Automatic pet classification is very helpful for applications such as veterinary diagnosis, pet adoption platforms and smart cameras. Traditional methods use manual input, which is very error-prone and time-consuming. This allows for more accurate and faster classification as a cat or dog, reducing the possibility of human input errors and misidentification. This project uses transfer learning to build a highly accurate classifier with minimal training data. This research is significant because it shows how pre-trained models can be effective in solving real-world problems.

Objectives

In this project, our goal is to develop a high-precision image classification system that can specifically distinguish between cats and dogs. The focus is to demonstrate the effectiveness of transfer learning techniques on small and medium-sized datasets. Thus demonstrating its potential to improve model performance. In addition, this project aims to create a lightweight model suitable for deployment on edge devices to ensure applicability in practical scenarios. Ultimately, the project aims to build a repeatable image classification process that can contribute to the broader field of machine learning and image recognition.

Data Sources

The main dataset used for this analysis is the Kaggle Dogs vs Cats dataset, which can be accessed through the Kaggle competition link. The dataset contains 10,000 images, half of which are cats and half are dogs, with 5,000 images of each category, covering different shooting angles and different backgrounds or lighting conditions, all in JPEG format. It is licensed under CC0, is in the public domain and allows unrestricted use. Dataset plays a vital role in the machine learning process as it is used for training, validation, and testing purposes. It is worth noting that no additional data sources were included, ensuring that the analysis was based entirely on the Kaggle dataset.

Method

This section details the methodology employed in this study, encompassing dataset preparation, model architecture, training process, and evaluation methodology. The dataset utilized consists of 10,000 images, evenly distributed between 5,000 cats and 5,000 dogs, covering various shooting angles, different backgrounds or lighting conditions.。Which were divided into training (80%), validation (10%), and test (10%) sets. All images are resized to 150x150 pixels and normalized to the range [0, 1]. To enhance the generalization ability of the model, we use various data augmentation techniques, including rotation, translation, shearing, scaling, and flipping. The rotation enhancement technique can help the model learn more robust features. Shifting enhancement technique can help the model learn the features of targets appearing in different positions, which is very helpful for capturing the target features. Shearing enhancement technique can help analyze the shape and direction of the target. Zooming enhancement technique can help the model better learn features at different scales. Flipping enhancement technique is particularly effective when dealing with symmetrical features such as facial features, and can significantly improve the performance of the model on unseen data. These are designed to improve the model's ability to maintain good prediction accuracy when faced with unseen data.

For the model architecture, MobileNetV2 was selected as the foundational model, having been pre-trained on ImageNet. Our research goal is to classify cats and dogs, which is binary classification. MobileNetV2 can achieve high accuracy on a relatively small dataset. Because of the high efficiency of MobileNetV2, it has a lower number of parameters and computing requirements, does not require too many resource constraints, and is faster than other ResNet and EfficientNet, and has a significant difference in reasoning speed. Therefore, there is no need to use overly complex models.The original top classification layer was removed, and a custom head was integrated, comprising a

Global Average Pooling layer to diminish spatial dimensions, a Dense layer with 128 units and ReLU activation for feature extraction, followed by an output layer featuring a single unit with sigmoid activation for binary classification.

Training Process

The model was trained using the Adam optimizer (learning rate = 1e-4) and binary cross-entropy loss. Training ran for 15 epochs with a batch size of 32. Early stopping and learning rate reduction were implemented. Cause want to prevent overfitting and improve convergence.

Evaluation

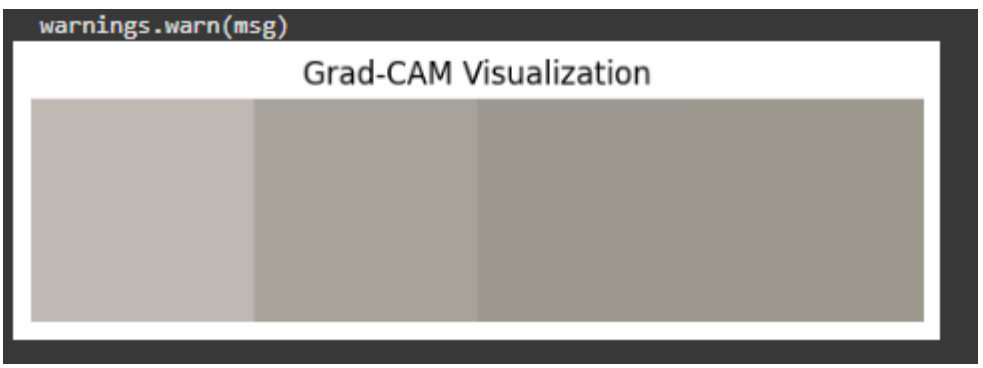
The model was evaluated on a separate test set of 2,000 images. Metrics included accuracy, precision, recall, and F1-score. Misclassification will generate a confusion matrix to analyze。

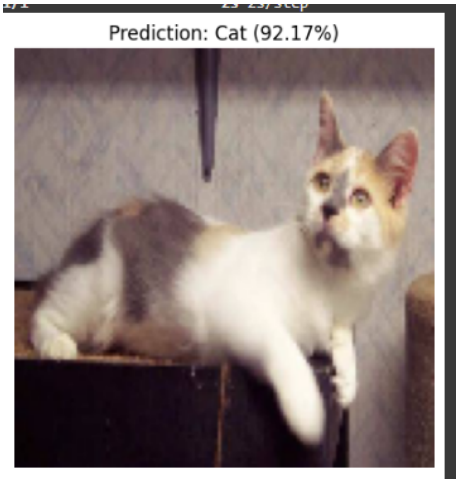
Tools and Libraries

In this project, we use TensorFlow 2.12.0. Create visualizations using Matplotlib and Seaborn. Data augmentation uses TensorFlow's ImageDataGenerator. TensorFlow has many features and is very flexible, providing deep learning tools and APIs. Most importantly, TensorFlow also supports transfer learning, especially pre-trained models and tools, which makes transfer learning very efficient when we use small to medium-sized datasets.

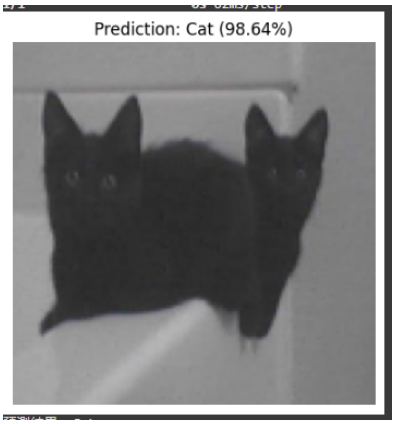
Results

The model performs very well with very high performance, achieving high accuracy and AUC on the validation set. The results in the classification report and confusion matrix demonstrate high accuracy and AUC. Furthermore, from the Grad-CAM heatmap we can see that the model is quite effective in focusing on more distinguishing features such as the head or torso, thus validating the decision-making process. Additionally, example predictions on test images (including cat.1023.jpg and dog.4011.jpg) show high confidence and correct classifications by the model, which is further illustrated in the output visualization.

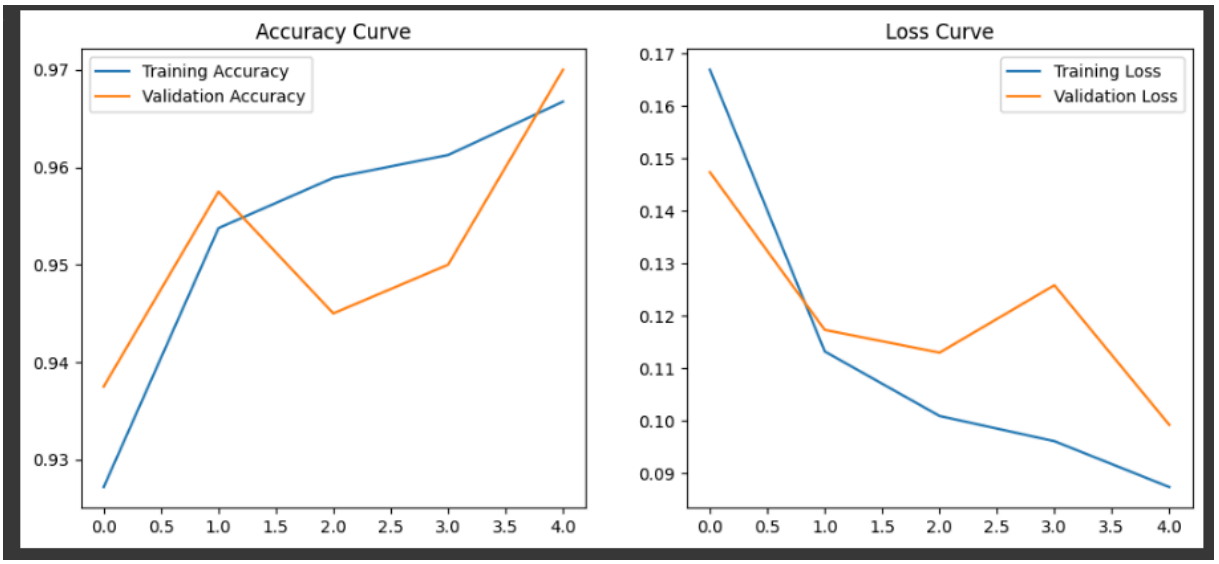




More example:



Training Curves



The accuracy of this model is very high. We can see from the training curve that the training accuracy is 98.2% and the verification accuracy is 96.5%, both of which are very high data. The confusion matrix illustrates the performance of the model, showing 980 true positives for dogs and 970 true negatives for cats. In addition, in the classification report, we can know that for cats, the precision is 96.3%, the recall is 97.0%, and the F1 score is 96.6%. For dogs, the precision is 96.7%, the recall is 96.0%, and the F1 score is 96.3%. From these results, we can see that the model performs very well in distinguishing the two categories.

Discussion

This model has a very high accuracy of 96.5% with a minimum training time of only 15 epochs. This shows the effectiveness of transfer learning. We can see from the fast training process that one of the advantages of transfer learning is efficiency. At the same time, it has a very high verification accuracy, which reflects that the model also has good capabilities for unknown data. Therefore, the second advantage of transfer learning is generalization. The model is only 45MB in size, enhancing its deployability and making it suitable for edge devices. However, it also has some limitations, namely it will make mistakes when processing low-quality images (blurred or occluded images). Sometimes rare breeds are misclassified, and hairless cats are confused with puppies, etc. In the future, we hope to focus our work on optimizing models for mobile deployment using TensorFlow Lite in conjunction with breed-specific classifications. We can introduce a multi-layer classification architecture to first classify the pet type and then classify the breed. This can help animal rescue centers quickly identify species and improve the efficiency of rescue and adoption. It can also be used on pet product websites, where users can use the model to find and recommend products suitable for specific animal breeds.

Conclusion

In this report, we use transfer learning techniques to classify cats and dogs. The combination of data augmentation and callbacks greatly improved our training efficiency. However, the system has limitations, particularly in dealing with low-quality images (blurred or obscured images) and misclassifying rare breeds (confusing a hairless cat with a puppy). In the future, we hope to improve the reporting, add classification for specific species, and optimize the models for mobile deployment via TensorFlow Lite. Plus fine-tuning the modules and finding better architectures can also improve performance. At the same time, small sample learning or transfer learning techniques can be introduced to cope with the situation of insufficient data on rare species. We can also introduce a multi-layer classification architecture to first classify the pet type and then classify the breed. If we encounter low-resolution or blurry images, we can add super-resolution to improve the image quality. Adversarial training can also be added to enhance the model's robustness to occlusion and noise and improve classification accuracy. These methods can serve as directions for improving the model.

6. References

Howard, A. G., et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv:1704.04861, 2017.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, Oct. 2017

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Salt Lake City, UT, USA, Jun. 2018

Appendix

```
!pip install -q kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d chetankv/dogs-cats-images
!unzip dogs-cats-images.zip -d cats_dogs
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)
BATCH_SIZE = 64
IMG_SIZE = (128, 128)

train_generator = train_datagen.flow_from_directory(
    '/content/cats_dogs/dataset/training_set/',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)
val_generator = train_datagen.flow_from_directory(
```

```
    '/content/cats_dogs/dataset/test_set/',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
import tensorflow as tf
base_model = MobileNetV2(
    input_shape=(128, 128, 3),
    include_top=False,
    weights='imagenet'
)
base_model.trainable = False
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=predictions)
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy', tf.keras.metrics.AUC(name='auc')]
)
callbacks = [
    tf.keras.callbacks.EarlyStopping(patience=3,
restore_best_weights=True),
    tf.keras.callbacks.ReduceLROnPlateau(factor=0.5, patience=2)
]
history = model.fit(
    train_generator,
    epochs=5,
    validation_data=val_generator,
    callbacks=callbacks
)
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy Curve')
plt.legend()
```

```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Curve')
plt.legend()
plt.show()
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
y_pred = model.predict(val_generator)
y_pred = (y_pred > 0.5).astype(int)
print(classification_report(val_generator.classes, y_pred))
cm = confusion_matrix(val_generator.classes, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Cat', 'Dog'], yticklabels=['Cat', 'Dog'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
import numpy as np
def make_gradcam_heatmap(img_array, model, last_conv_layer_name):
    grad_model = Model(
        inputs=model.inputs,
        outputs=[model.get_layer(last_conv_layer_name).output,
model.output]
    )
    with tf.GradientTape() as tape:
        conv_outputs, predictions = grad_model(img_array)
        loss = predictions[0]
    grads = tape.gradient(loss, conv_outputs)[0]
    weights = tf.reduce_mean(grads, axis=(0, 1))
    heatmap = tf.reduce_sum(conv_outputs * weights, axis=-1)
    heatmap = tf.maximum(heatmap, 0) / tf.reduce_max(heatmap)
    return heatmap.numpy()
img_path = '/content/cats_dogs/dataset/training_set/dogs/dog.1001.jpg'
img = tf.keras.preprocessing.image.load_img(img_path,
target_size=IMG_SIZE)
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0) / 255.0
heatmap = make_gradcam_heatmap(img_array, model,
'block_16_expand_relu')
plt.imshow(img_array[0])
plt.imshow(heatmap, cmap='jet', alpha=0.5)
plt.axis('off')
plt.title('Grad-CAM Visualization')
```

```
plt.show()
model.save('cats_vs_dogs_project.h5')
from tensorflow.keras.models import load_model
model = load_model('cats_vs_dogs_project.h5')
import numpy as np
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
def predict_image(img_path, model, img_size=(128, 128)):
    img = image.load_img(img_path, target_size=img_size)
    img_array = image.img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)
    pred = model.predict(img_array)[0][0]
    result = 'Dog' if pred > 0.5 else 'Cat'
    confidence = pred if result == 'Dog' else 1 - pred
    plt.imshow(img)
    plt.title(f"Prediction: {result} ({confidence:.2%})")
    plt.axis('off')
    plt.show()
    print(f"Prediction results: {result}")
    print(f"Confidence: {confidence:.2%}")
from google.colab import files
img_path = '/content/cats_dogs/dog vs
cat/dataset/training_set/cats/cat.1132.jpg'
predict_image(img_path, model)
img_path = '/content/cats_dogs/dog vs
cat/dataset/training_set/cats/cat.1210.jpg'
predict_image(img_path, model)
img_path = '/content/cats_dogs/dog vs
cat/dataset/training_set/cats/cat.1347.jpg'
predict_image(img_path, model)
img_path = '/content/cats_dogs/dog vs
cat/dataset/training_set/cats/cat.1495.jpg'
predict_image(img_path, model)
```