Report of Exercise 1 – Image classification

1. Pipeline
    a. Data loading and validation

The dataset was downloaded from Kaggle (Microsoft Cats vs Dogs Dataset). Invalid images were filtered out by checking their format and structure.

    b. Data splitting

The data was divided into training, validation, and testing sets. Only valid images were included, and the dataset was balanced between the two classes.

    c. Data augmentation

Applied ImageDataGenerator to perform augmentation techniques such as rotation, resizing, horizontal flipping, etc. Preprocessed the images using the preprocess_input function from VGG16 to normalize input data.

    d. Model development

The VGG16 model (pre-trained on ImageNet) was used as the backbone.

Added custom layers, including:

- Dense layers with Batch Normalization to enhance convergence.
- Dropout layers to reduce overfitting.
- Only the custom layers were fine-tuned, while the VGG16 layers were frozen.

    e. Training and evaluation

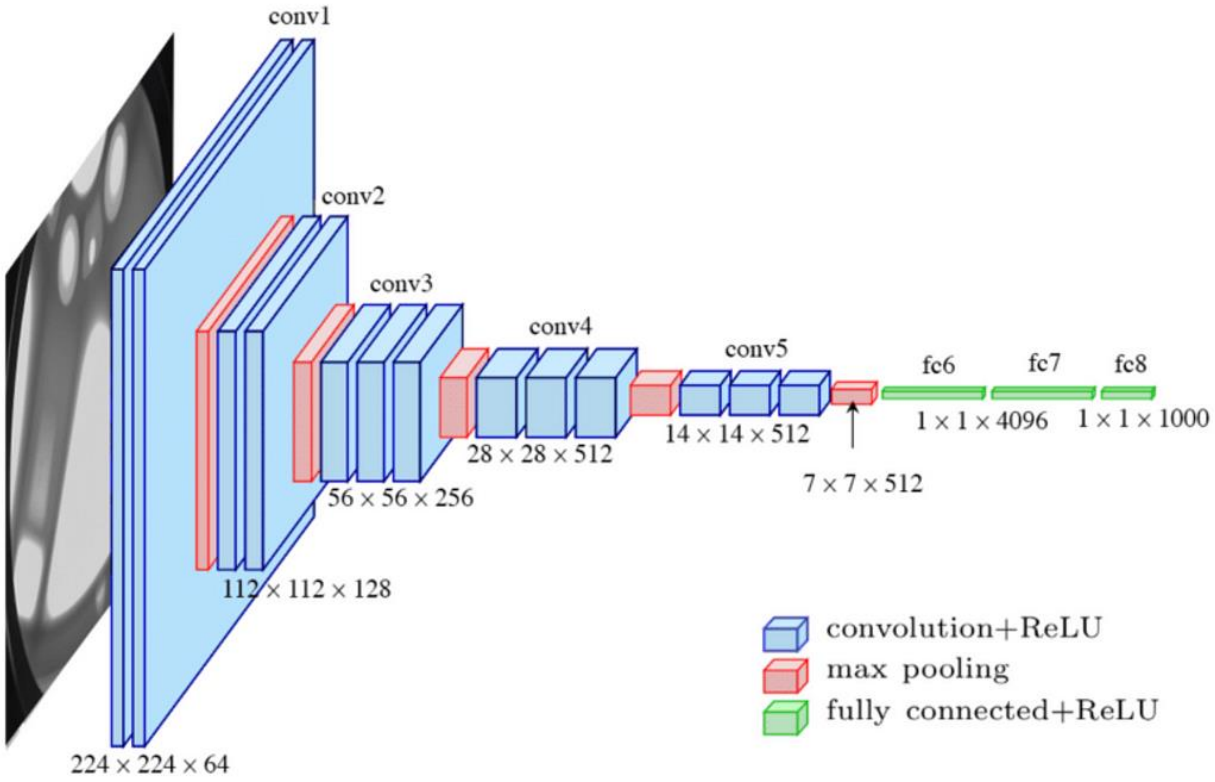
The model was trained using the training and validation datasets. Utilized callbacks like ModelCheckpoint to save the best model and ReduceLROnPlateau to adjust the learning rate when necessary. Evaluated the model on the test set and visualized the results (accuracy and loss).

    f. Deployment

The trained model was deployed as a web application using Flask. Users can upload an image through the web interface, and the application predicts whether the image contains a cat or a dog. The result is displayed along with the uploaded image for visualization.

2. Algorithms used:

VGG16: A robust CNN model for deep feature extraction from images.



Using Binary Crossentropy loss and Adam optimizer with adaptive learning rates

3. Remaining issues

- Some images have inconsistent sizes or lighting conditions, impacting model performance.
- The model may overfit the training data, even with data augmentation.
- Flask is lightweight and suitable for initial deployment, but scaling for multiple concurrent users may require a more robust framework.

4. Improvement ideas

Augment the dataset with additional images or apply more advanced augmentation techniques (e.g., brightness and color adjustments).

Fine-tune the model to adapt better to the specific features of this problem. Experiment with models like MobileNet or EfficientNet for faster training with comparable performance.

Use other metrics such as Precision, Recall, F1-score for more comprehensive evaluation, especially when there is data imbalance.

Deploy the model using a more scalable platform such as FastAPI or integrate it into a cloud-based service like AWS, GCP, or Azure for handling higher traffic.