Dog Emotion Detection

Wen-Chi Lee, Jeansue Wu, Tzuliang Huang, Xinyang Zhou

Problems

- **Object Detection:** Detecting and classifying objects (dogs and cats) in images.
  - CNN vs. InceptionV3
- **Emotion Classification:** Classifying dog emotions (happy, angry, sad, relaxed) in images.
  - VGG16 vs. MobileNetV2

Object Detection | **Emotion Classification**

Detected to be a dog | Detected to be a sad dog

Objective

To evaluate the impact of data filtering on model accuracy by leveraging object detection to enhance the quality of our dataset.

Datasets:

- **Dog and Cat Dataset:** Images of dogs and cats for training the object detection model.
- **Dog Emotion Dataset:** Images labeled with dog emotions (happy, angry, sad, relaxed), containing both dog and non-dog data.

Methodology

Step 1

Train **Object Detection** Model using the dog and cat dataset.

Step 2

Filter Dog Emotion Dataset: Apply the object detection model to remove non-dog images.

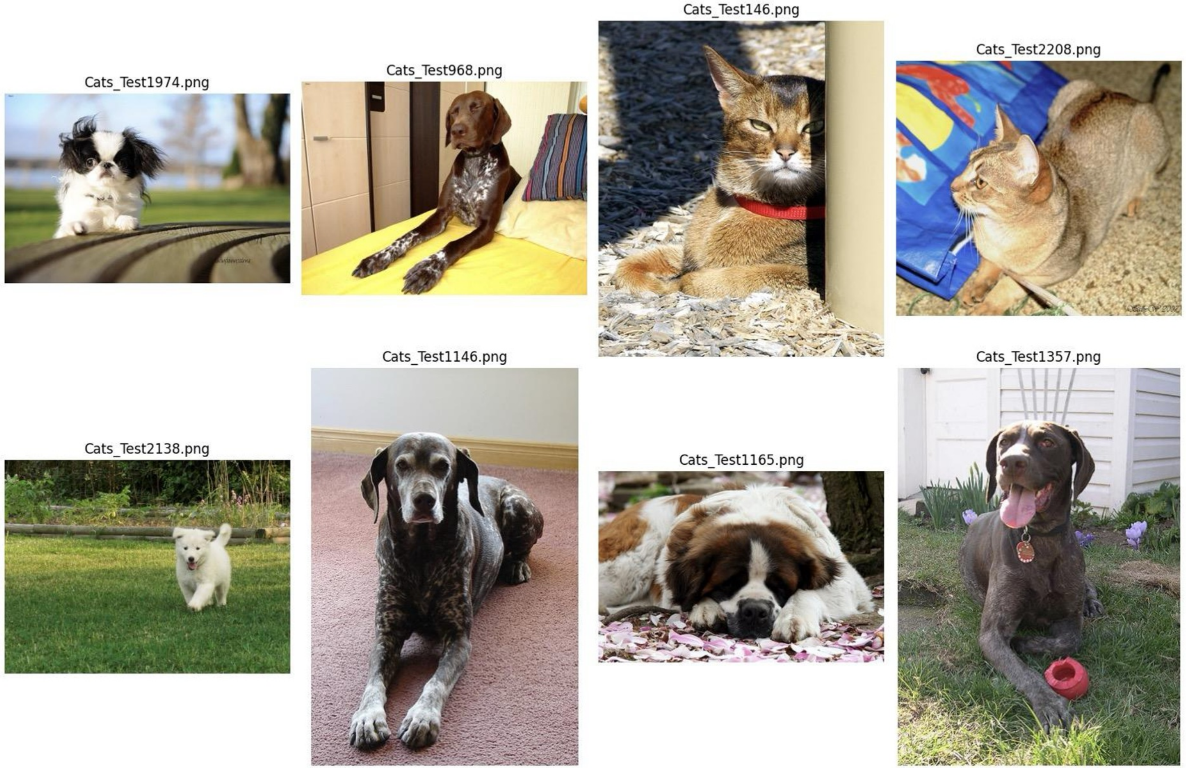
Step 3

Train **Emotion Classification** Models using VGG and MobileNet on both the original and filtered datasets.

Step 4

Compare Model Performance by evaluating and comparing accuracy metrics.

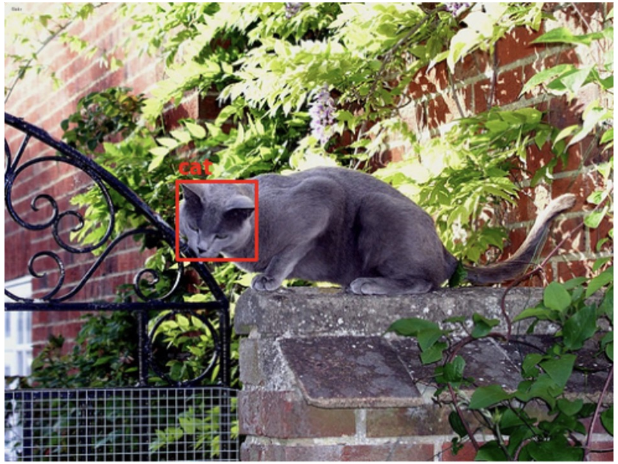
Dog and Cat Dataset - Images



Dog and Cat Dataset - Annotations

```
{'size': (500, 375),
 'filename': 'Cats_Test100.png',
 'class': 'cat',
 'bbox': [142, 145, 206, 209]}
```

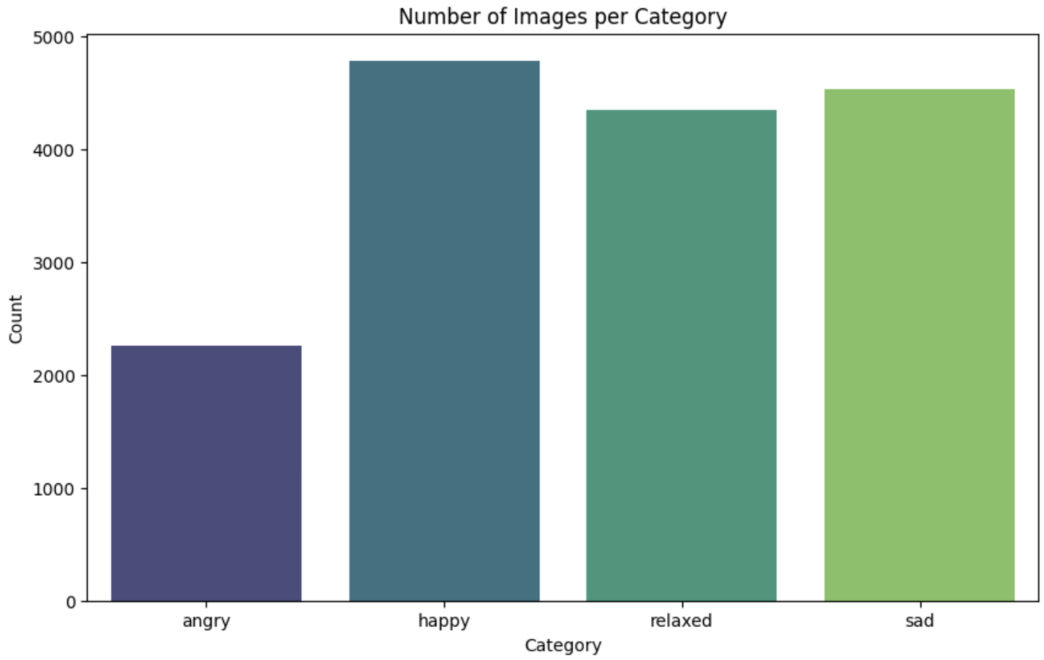
Cats_Test100.png



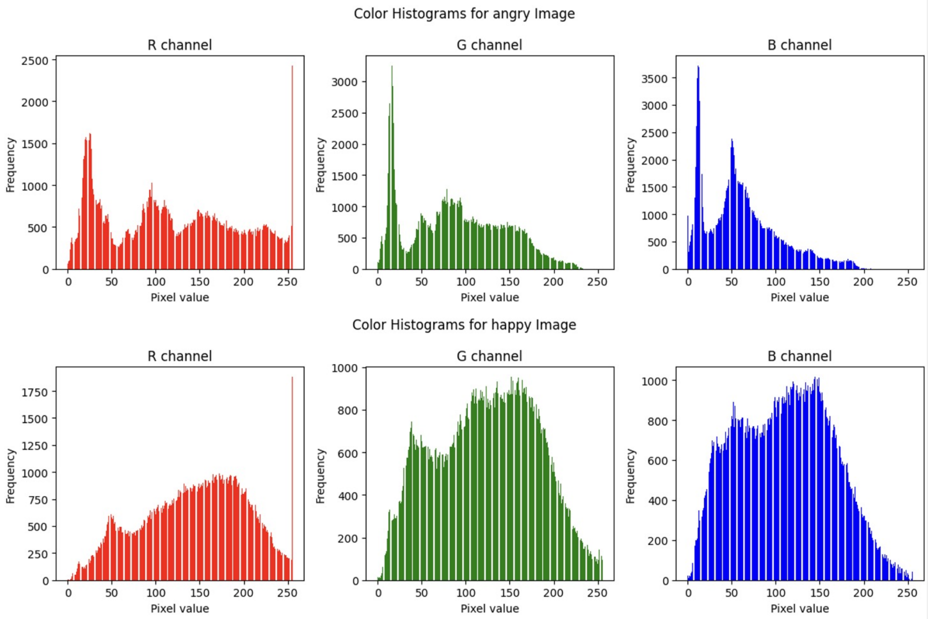
Dog Emotion Dataset

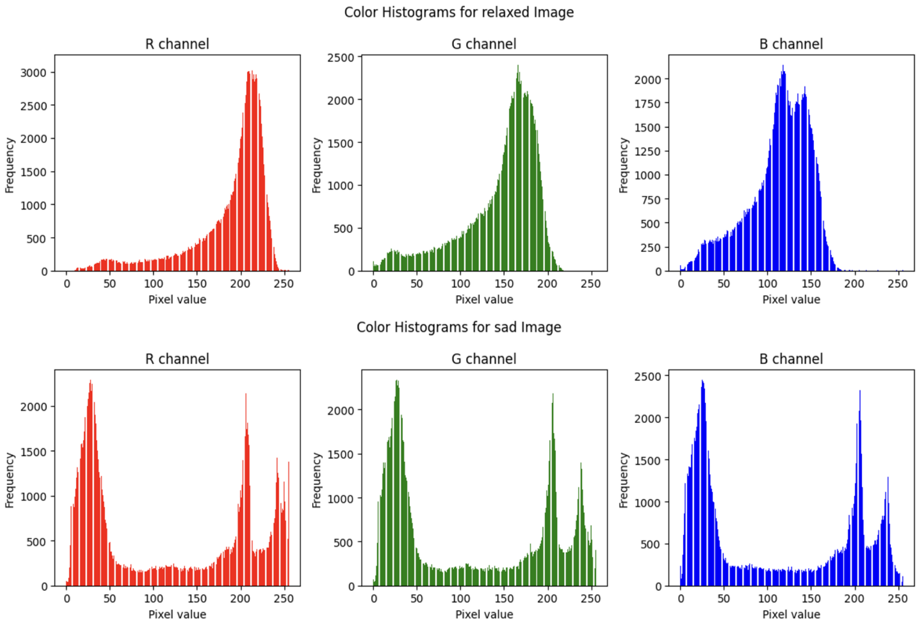
EDA

```
Emotion: angry
  Record Count: 2256
  Missing Values: 0

Emotion: sad
  Record Count: 4532
  Missing Values: 0

Emotion: happy
  Record Count: 4784
  Missing Values: 0

Emotion: relaxed
  Record Count: 4349
  Missing Values: 0
```



Color Histogram





Object Detection Model - Customize CNN

Preprocessing:

- Stack images into a batch and pass through the **pre-trained ResNet50 model** to obtain **feature representations**
- Convert class labels to one-hot encoded vectors and normalize bounding box coordinates

Model Architecture:

- Input: (7, 7, 2048)
- Layers:
  - Conv2D (32 filters, 3x3 kernel, ReLU activation)
  - MaxPooling (2x2 pool size, same padding)
  - Conv2D (64 filters, 3x3 kernel, ReLU activation)
  - MaxPooling (2x2 pool size, same padding)
  - Conv2D (128 filters, 3x3 kernel, ReLU activation)
  - MaxPooling (2x2 pool size, same padding)
  - Flatten
  - Dense (128 units, ReLU activation)
- Outputs:
  - Dense (2 units, softmax activation, named 'head_classes')
  - Dense (4 units, named 'head_boxes')

Object Detection Model - Customize CNN result

- Train:
  - Classification:
    - Accuracy: 100%
    - Loss: 1.1617e-06
  - Bounding Box:
    - Loss: 362.6393
  - Classification + Bounding Box accuracy: 0.693

- Valid:
  - Classification:
    - Accuracy: 99.7%
    - Loss: 0.0127
  - Bounding Box:
    - Loss: 369.1981
  - Classification + Bounding Box accuracy: 0.228

Object Detection Model - InceptionV3

- Purpose: Efficient, deep convolutional neural network for image classification and object detection
- Key Features:
  - Multi-scale feature extraction
  - Reduced computational cost
  - Deep layers for complex pattern learning

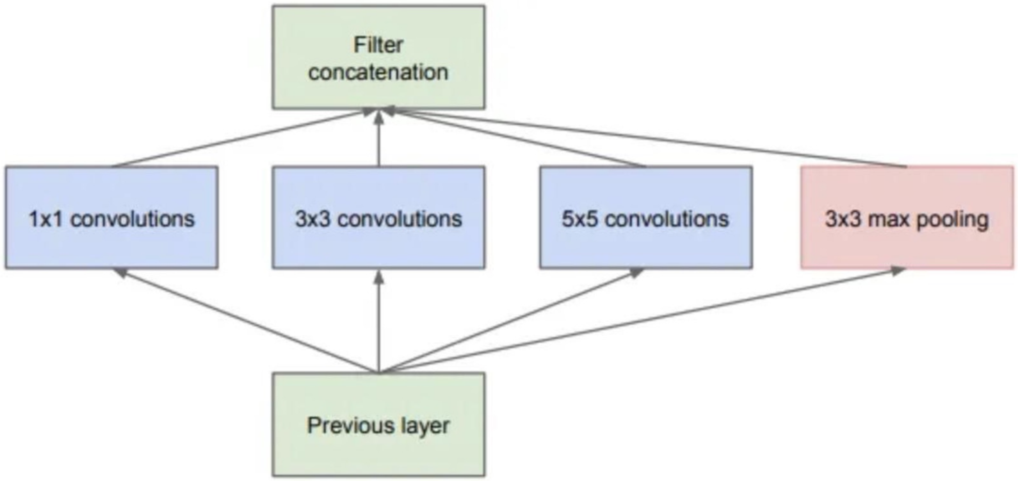
Object Detection Model - InceptionV3

Core Idea: Parallel application of multiple convolutional and pooling operations

Components:

- 1x1, 3x3, and 5x5 convolutions
- Max pooling

Benefits: Captures different types of features, reduces computation



Object Detection Model - InceptionV3 result

Training Metrics:

- Total Loss: 0.0038
- Bounding Box Regression Loss: 0.0055
- Classifier Loss: 0.0013
- Bounding Box Regression Accuracy: 84.80%
- Classifier Accuracy: 100.00%

Validation Metrics:

- Total Validation Loss: 0.0109
- Validation Bounding Box Regression Loss: 0.0043
- Validation Classifier Loss: 0.0208
- Validation Bounding Box Regression Accuracy: 83.67%
- Validation Classifier Accuracy: 99.27%

Model Comparison

- Classification:

  In terms of accuracy, both models perform similarly on classification tasks.

  (Customize CNN: 99.7% / InceptionV3: 99.27%)

- Bounding Box:

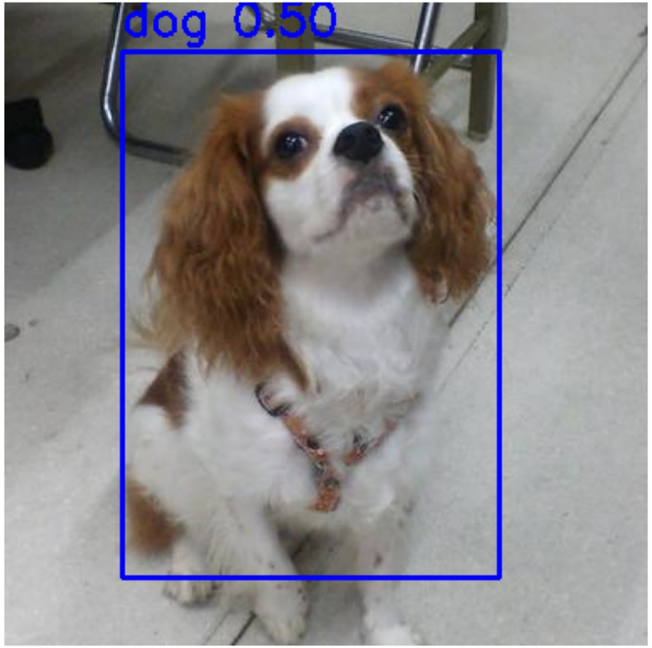
  InceptionV3 is more adept at accurately predicting the coordinates of the bounding boxes around objects in the images

  (Customize CNN: 369.1981 / InceptionV3: 0.0043)

Result

- Comprehensive and balanced dataset with no class imbalance.
- Enhancing Object Detection:
  - Fine-tuning
  - Other models (YOLO, RCNN)

YOLOv5 Detection with Bounding Boxes

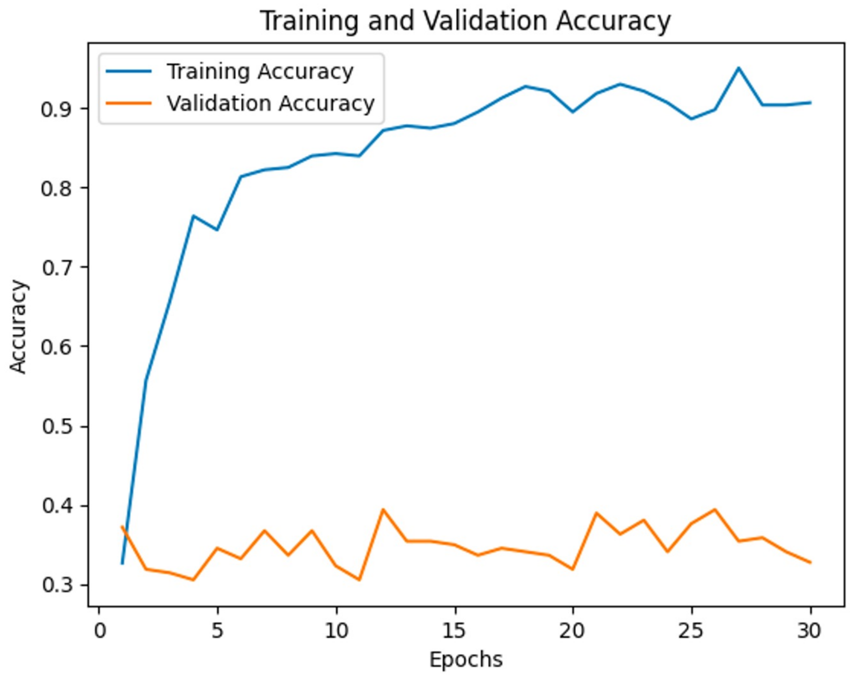


Emotion Classification Model - MobileNetV2 Structure

```
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(96, 96, 3))

# Add custom top layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu', kernel_regularizer=l2(0.01))(x)
x = BatchNormalization()(x)  # Add batch normalization
x = Dropout(0.3)(x)
x = Dense(256, activation='relu', kernel_regularizer=l2(0.01))(x)
x = BatchNormalization()(x)
x=Dropout(0.1)(x)
predictions = Dense(4, activation='softmax')(x)  # 4 classes for the 4 dog emotions

# Define learning rate scheduler
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=5,
    min_lr=0.00001
)

# Define early stopping
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)
```

- Customize layers
- Prevent overfitting (Batch Normalization, Dropout)
- Softmax for multiclass
- Learning rate monitor
- Early stopping

Emotion Classification Model - MobileNetV2 Accuracy



- Accuracy fluctuation
- Validation accuracy stays around 0.3-0.4
- Training accuracy increases to above 0.9

Emotion Classification Model - MobileNetV2 Metrics

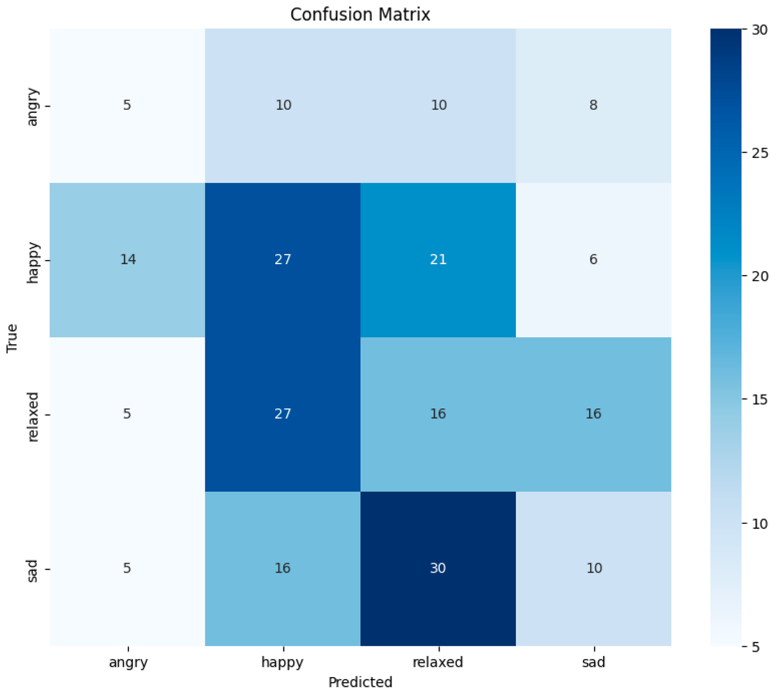
ROC AUC: 0.5032

Precision: 0.2431

Recall: 0.2522

F1 Score: 0.2264

Confusion Matrix:



Emotion Classification Model - VGG Structure

```python
# Load the VGG16 model, excluding the top layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))

# Add custom layers on top of VGG16
x = base_model.output
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
predictions = Dense(len(categories), activation='softmax')(x)

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer=Adam(lr=0.0001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Define early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=20,
    callbacks=[early_stopping]
)
```

Architecture: VGG16 pre-trained on ImageNet

Modification: Top layers removed, custom Layers Added

- Flatten: Converts 3D output to 1D vector
- Dense: 256 units, ReLU activation

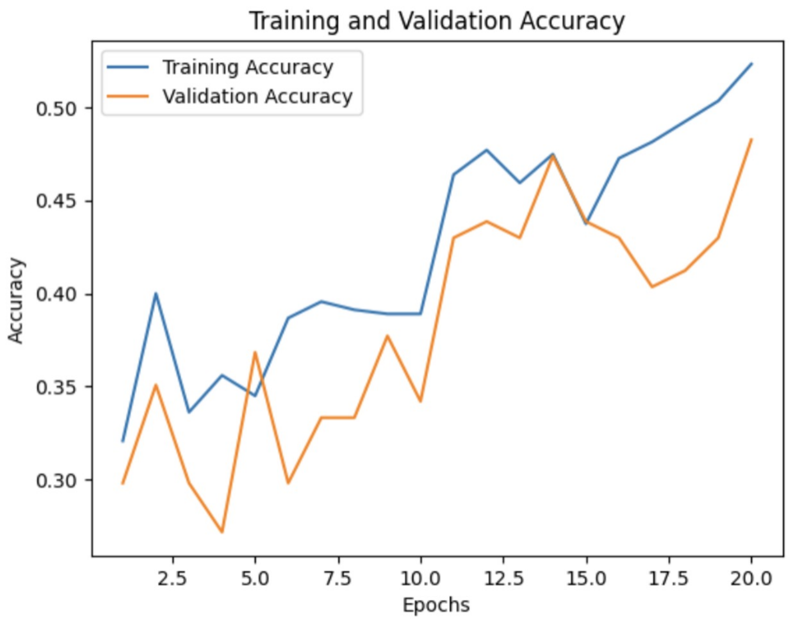
Training Configuration

- Optimizer: Adam with a learning rate of 0.0001
- Metric: Accuracy

Early Stopping

- Criteria: Stop if no improvement for 5 epochs

Emotion Classification Model - VGG Accuracy



Training Accuracy:

- Starts at approximately 0.30, shows fluctuations, and gradually increases to around 0.52 by the 20th epoch.

Validation Accuracy :

- starts at approximately 0.30, fluctuates more than the training accuracy, and ends at around 0.45 by the 20th epoch.

Emotion Classification Model - VGG Metrics

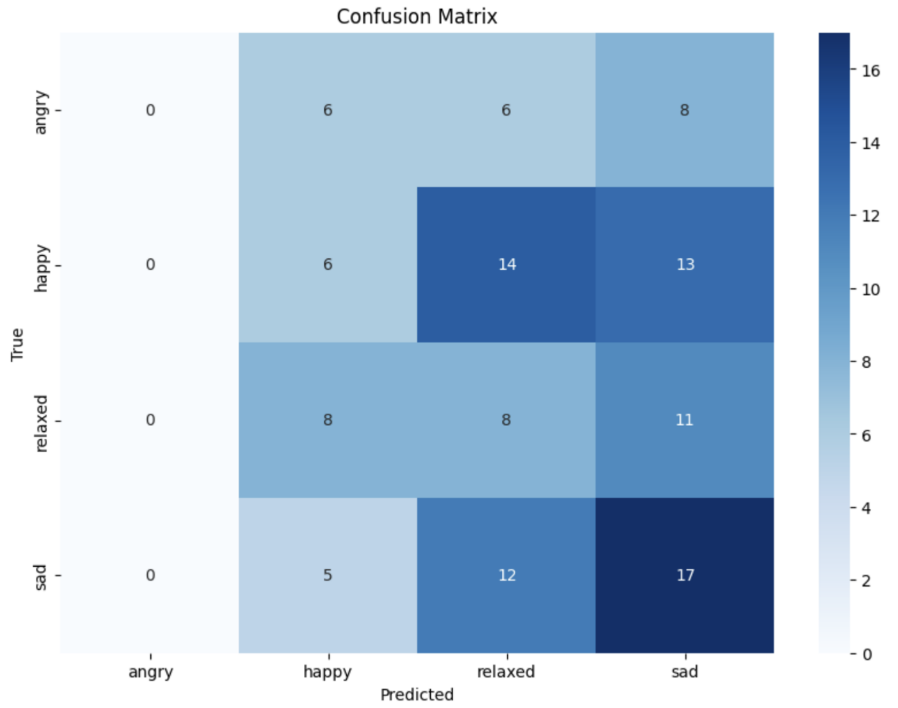
ROC AUC: 0.54

Precision: 0.24

Recall: 0.32

F1 Score: 0.27

Confusion Matrix:



Model Comparison

- VGG model outperformed the MobileNet model based on the metrics
- However, there is no image being classified as “angry” for the VGG model, which indicates a potential limitation of the model to recognize that emotion
- We trained the same model for the **Dog and Cat Dataset** and got a similar performance. There is no obvious improvement in emotion classification.

Results

- Some emotions are hard to classify (e.g. relaxed vs. sad). Image labels are also subjective.
- A bigger dataset with universal labeling standards may generate a better performance.
- If certain emotions are underrepresented in the dataset, it can lead to biased predictions. Ensuring a balanced representation of all classes can help improve the overall accuracy of the model.

Thank you!