Face Emotion Classification

Ravi shankar Purushothaman

What we have done so far:

- Building a CNN: A Convolutional Neural Network (CNN) from scratch to classify facial expressions into predefined emotion categories.

- RESNET50 Model: Implementing the RESNET50 architecture, pre-trained on ImageNet, to apply transfer learning for emotion recognition tasks.

-EfficientNetV2L Models: Implementing the EfficientNetV2L architecture, a state-of-the-art convolutional neural network known for its efficiency and effectiveness in image classification tasks. Fine-tuning model to improve the overall accuracy of val.

Data Preprocessing:

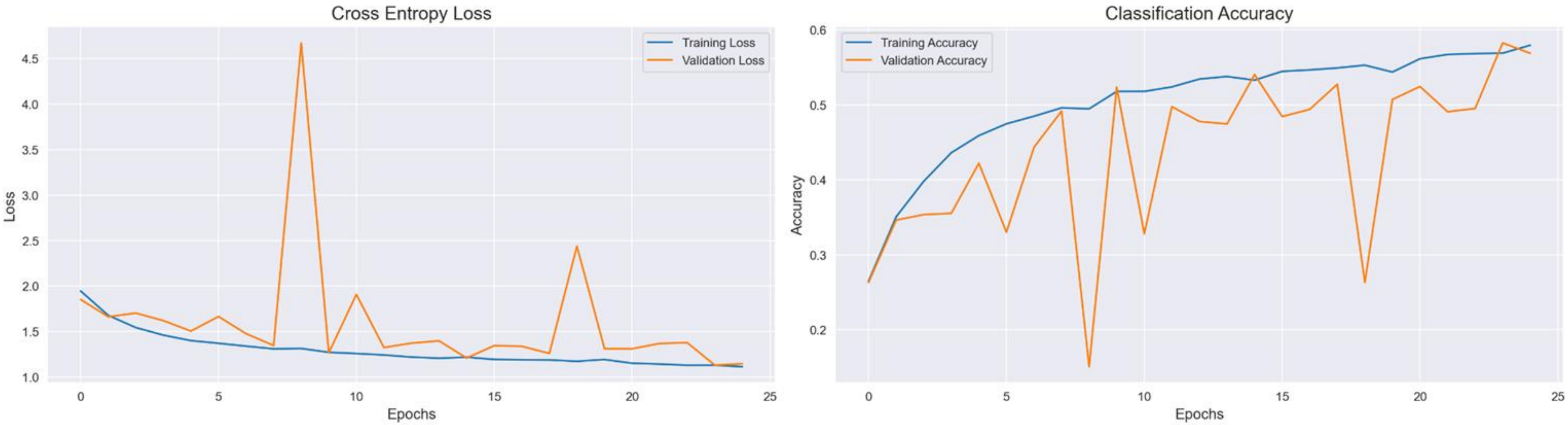
Data Generators for Training: Data Augmentation

- The dataset used in this project consists of facial images annotated with emotion labels.
- Each image is categorized into one of several emotion classes, including 'happy', 'sad', 'angry', 'surprise', 'fear', 'disgust', and 'neutral'.
- The dataset is preprocessed to standardize image sizes and formats, ensuring compatibility with the chosen deep-learning models.

- CNN from Scratch

```python
def define_model(image_size, image_channel):
    """
    Define a Convolutional Neural Network (CNN) model for image classification.
    """
    model = Sequential()

    # Input Layer
    model.add(Input(shape=(image_size[0], image_size[1], image_channel)))  # Input layer to specify

    # Convolutional Layers
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(256, (3, 3), activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    # Fully Connected Dense Layers
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.2))

    # Output layer
    model.add(Dense(NB_CLASSES, activation='softmax'))

    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

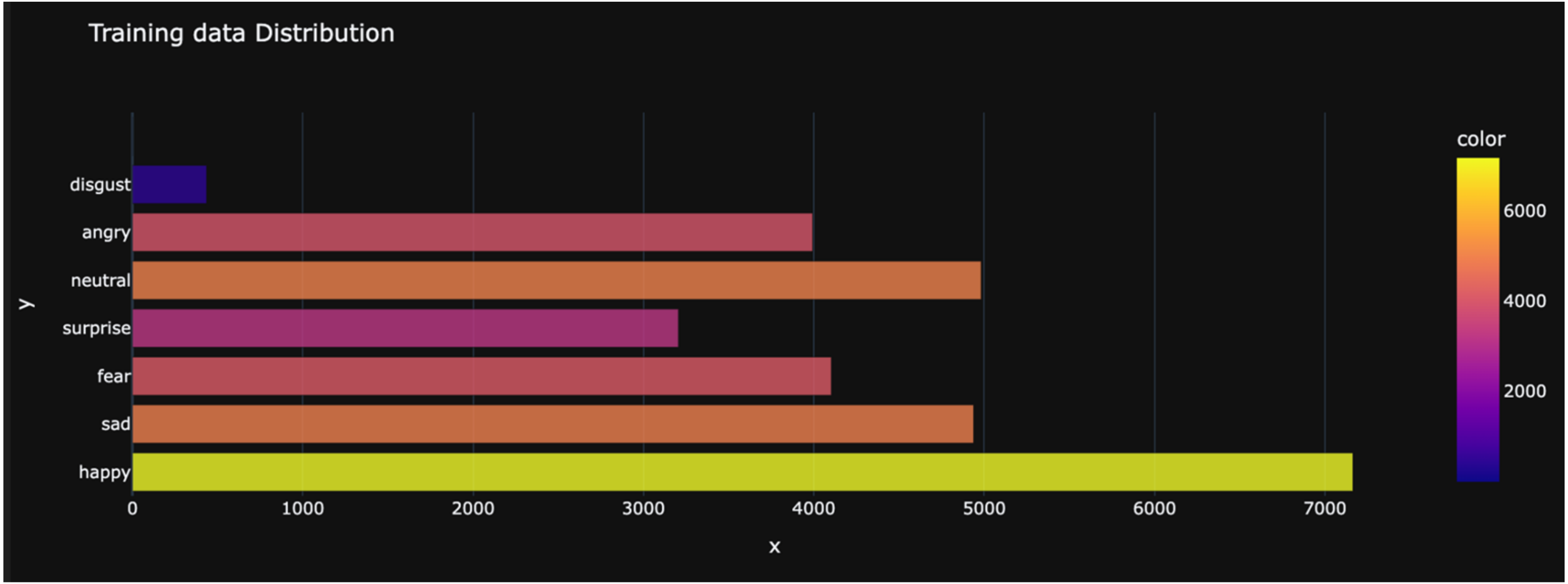
CNN Training Results:



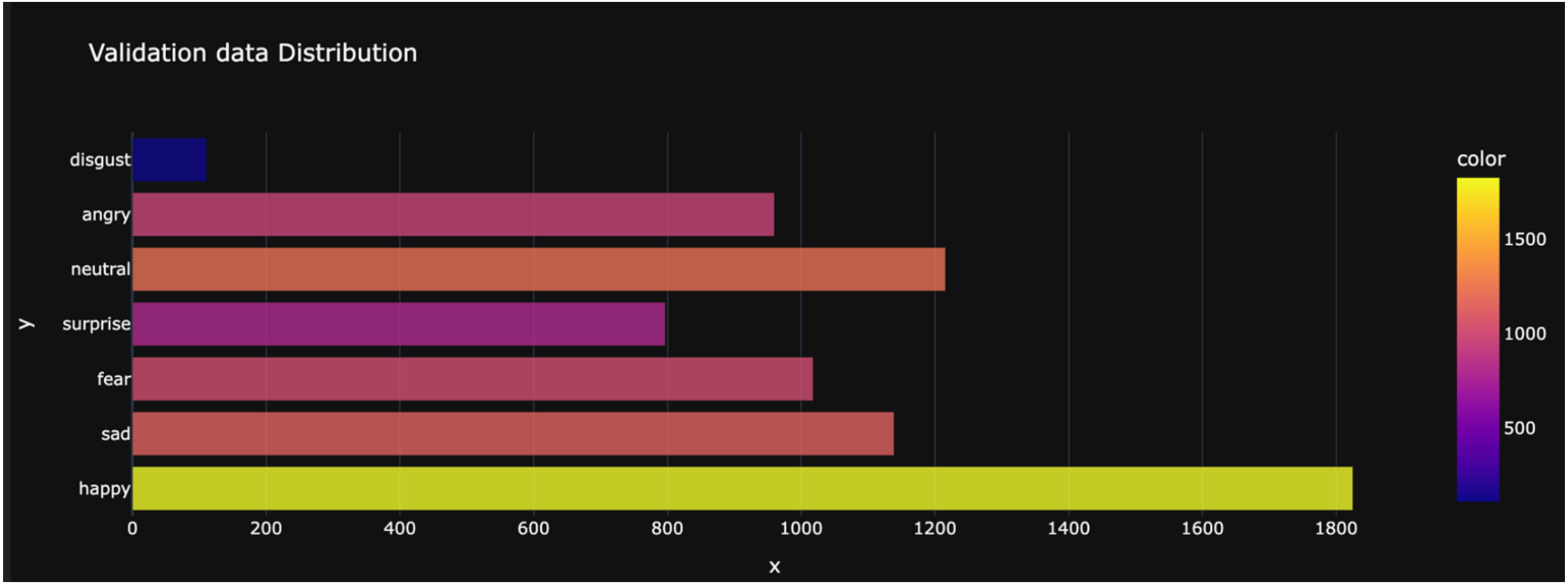
loss: 1.1122 - accuracy: 0.5796 - val_loss: 1.1455 - val_accuracy: 0.5688

CNN Eval Results (Test):

Classes are unbalanced?



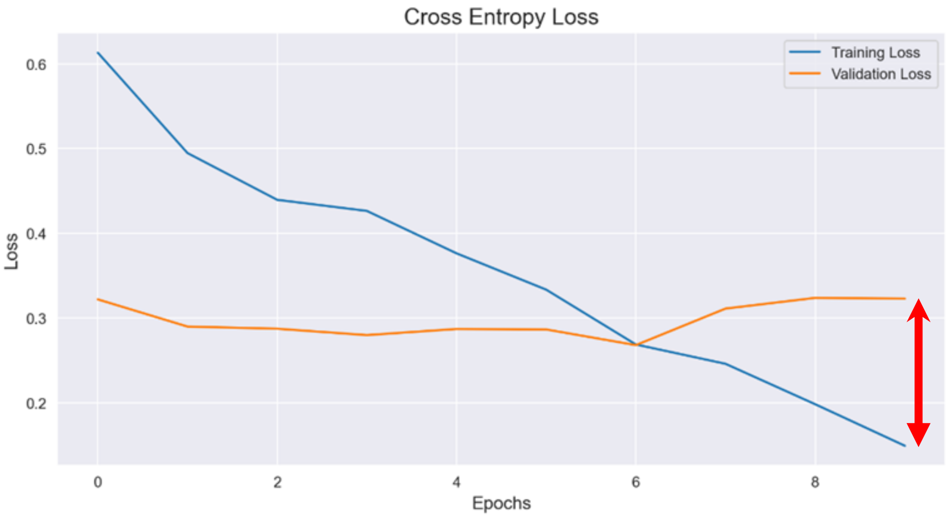
Classes are unbalanced?

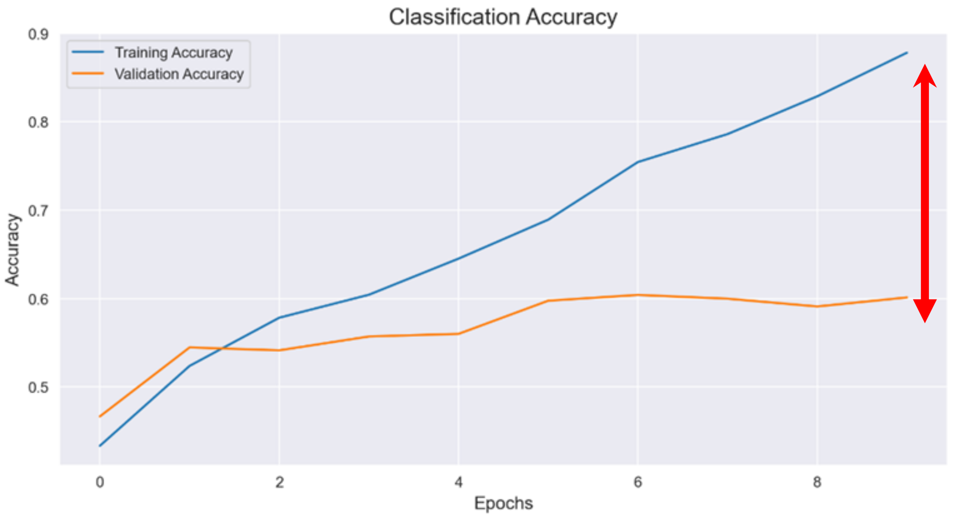


RESNET (50) - Applying Transfer Learning

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2, 2, 2048)        23587712

 global_max_pooling2d (Globa  (None, 2048)             0
 lMaxPooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 7)                 14343

=================================================================
Total params: 23,602,055
Trainable params: 23,548,935
Non-trainable params: 53,120
_________________________________________________________________
```

RESNET 50 Training Results:

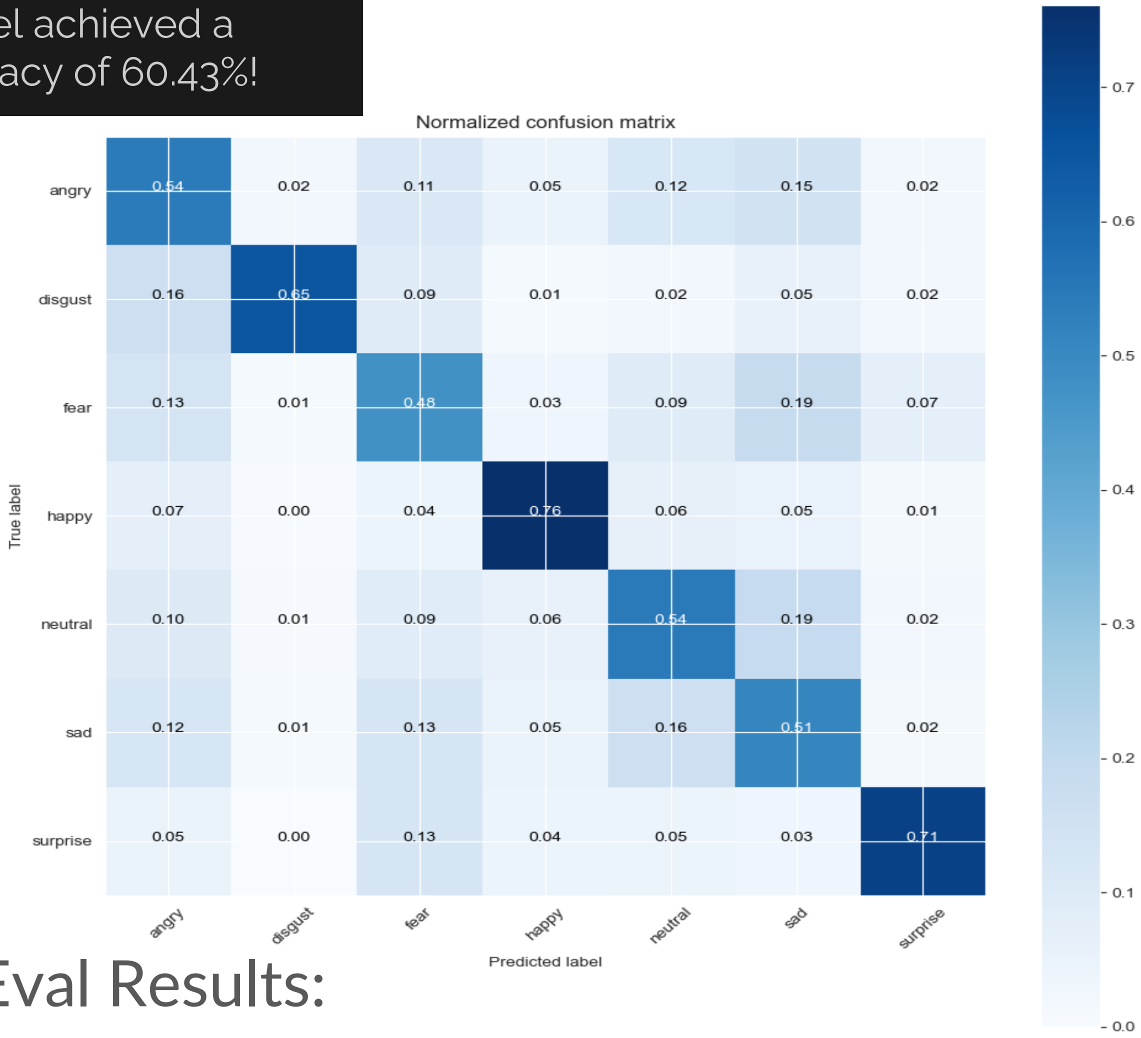




Loss: 0.1493, Accuracy: 0.8780 Validation Loss: 0.3230, Validation Accuracy: 0.6013

RESNET 50 Eval Results:

The current model achieved a categorical accuracy of 60.43%!



EfficientNetV2L

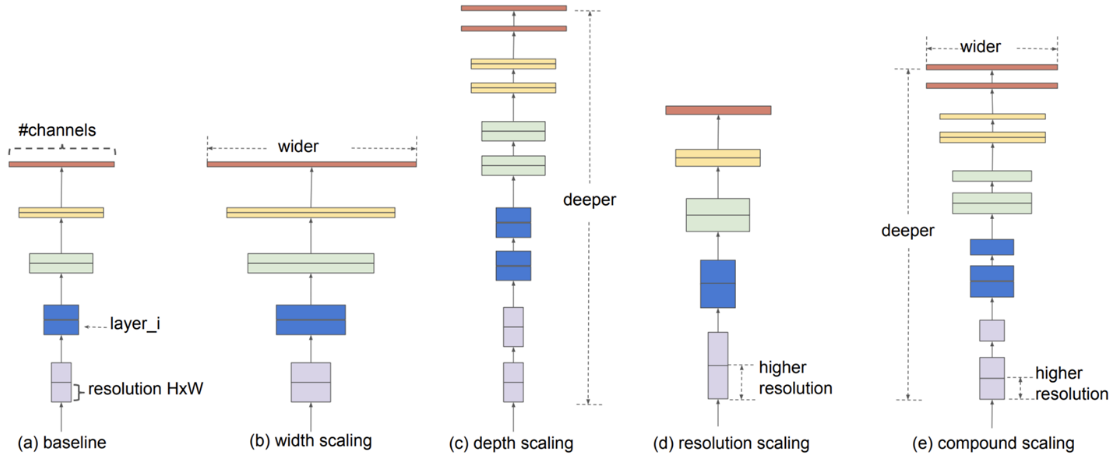
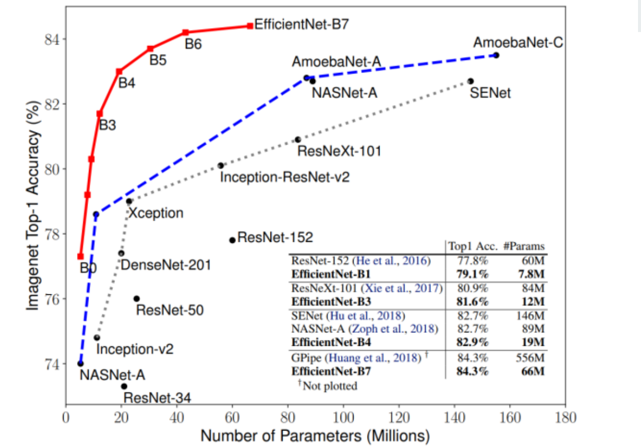
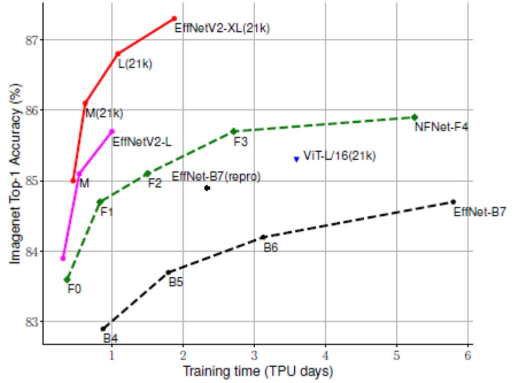


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

EfficientNetV2 is a type convolutional neural network that has faster training speed and better parameter efficiency than previous models. To develop these models, the authors use a combination of training-aware neural architecture search and scaling, to jointly optimize training speed.





(a) Training efficiency.

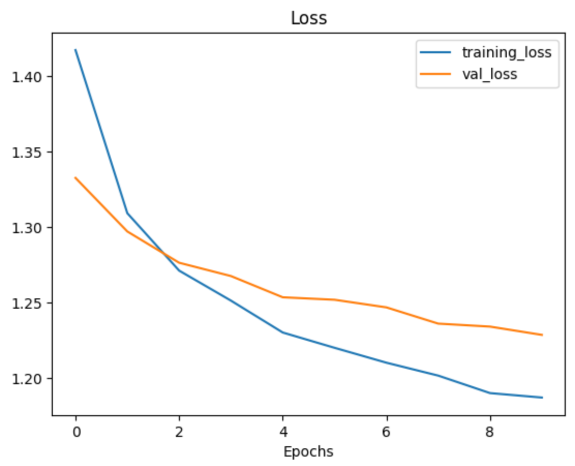
	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

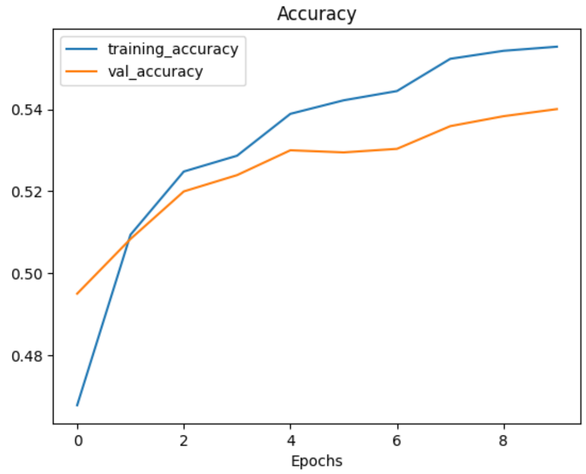
(b) Parameter efficiency.

EfficientNetV2L - 1)

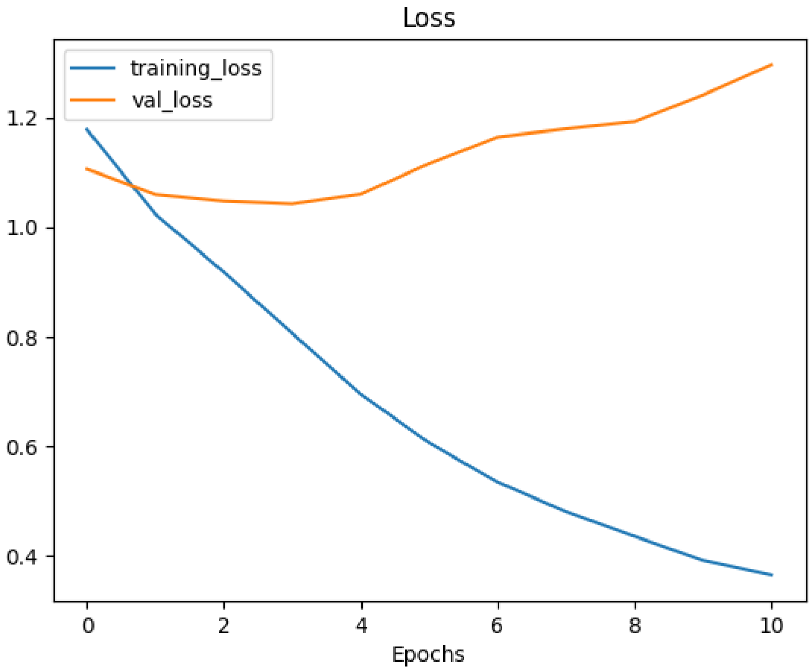
```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inpute_layer (InputLayer)   [(None, 224, 224, 3)]     0

 efficientnetv2-l (Function  (None, None, None, 1280   117746848
 al)                         )

 global_avrage_pooling (Glo  (None, 1280)              0
 balAveragePooling2D)

 Output_layer (Dense)        (None, 7)                 8967

=================================================================
Total params: 117755815 (449.20 MB)
Trainable params: 8967 (35.03 KB)
Non-trainable params: 117746848 (449.17 MB)
```

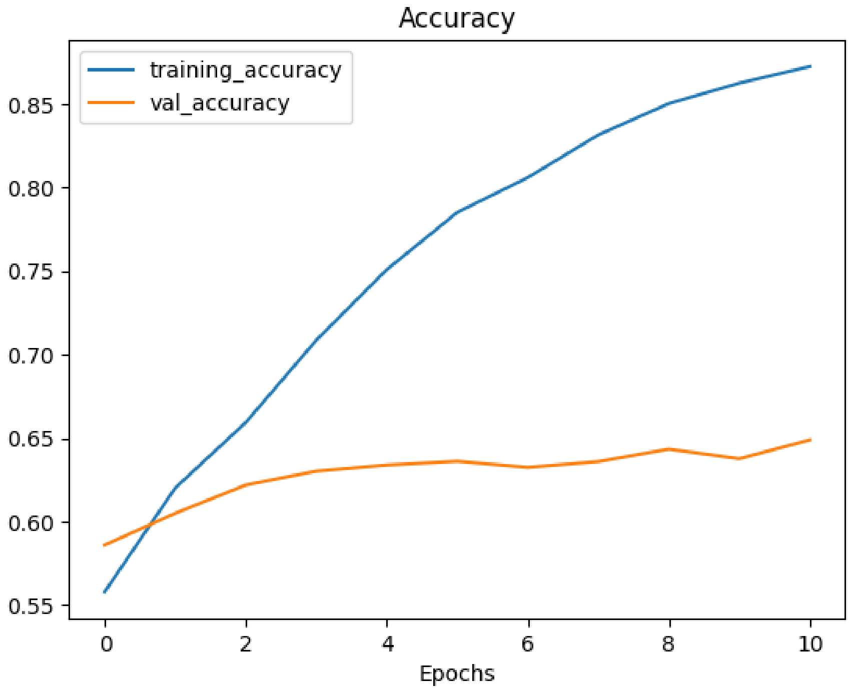
```
loss: 1.1869 - accuracy: 0.5553 - val_loss: 1.2284
- val_accuracy: 0.5400
```





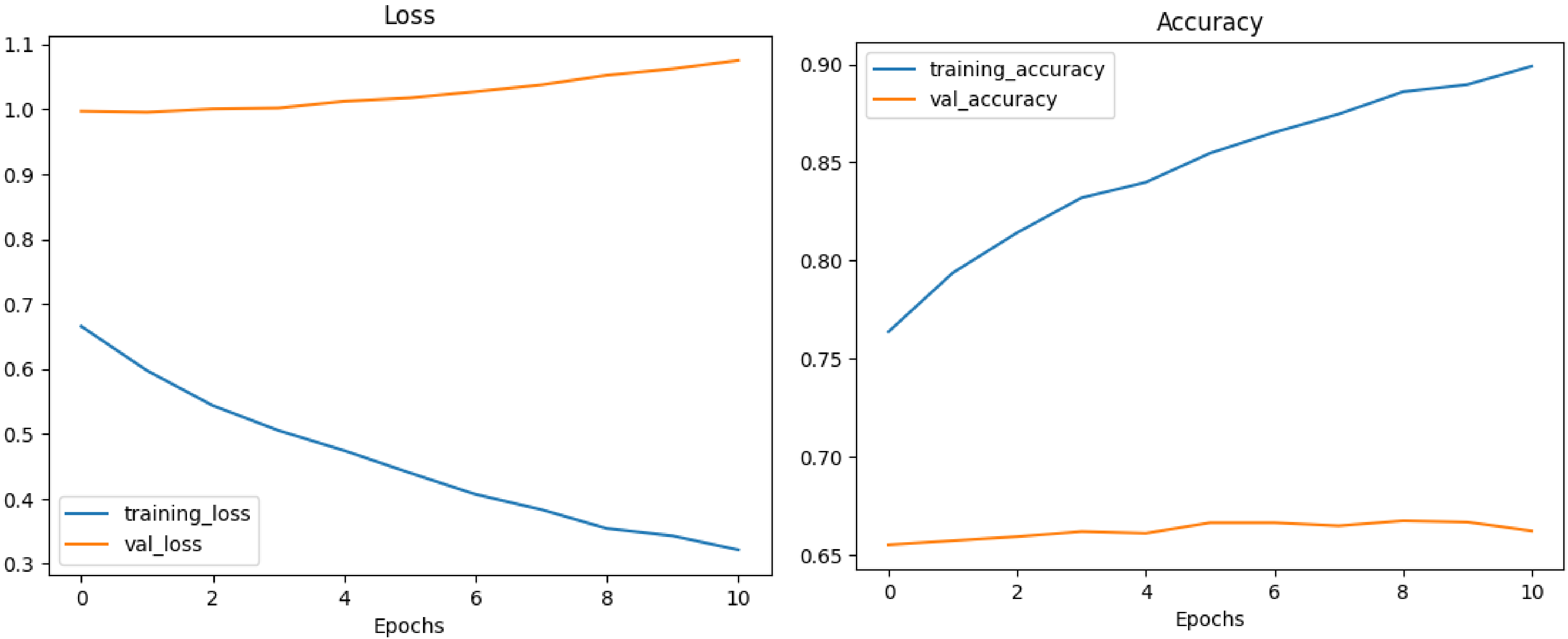
EfficientNetV2L - Unfreeze Last 20 Layers






```
accuracy: 0.8724 - val_loss: 1.2964 - val_accuracy: 0.6488
```

EfficientNetV2L - Unfreeze Last 45 Layers



loss: 0.3212 - accuracy: 0.8990 - val_loss: 1.0750 - val_accuracy: 0.6622

Model Performance Comparison & Conclusion

CNN
Loss: 1.1122, Accuracy: 0.5796
Validation Loss: 1.1455, Validation Accuracy: 0.5688

RESNET50
Loss: 0.1493, Accuracy: 0.8780
Validation Loss: 0.3230, Validation Accuracy: 0.6013

Base EfficientNetV2L
Loss: 1.1869, Accuracy: 0.5553
Validation Loss: 1.2284, Validation Accuracy: 0.5400

EfficientNetV2L (Unfreeze last 20 layers)
Loss: 0.3642, Accuracy: 0.8724
Validation Loss: 1.2964, Validation Accuracy: 0.6488

Efficient V2L (Unfreeze last 45 layers)
Loss: 0.3212, Accuracy: 0.8990
Validation Loss: 1.0750, Validation Accuracy: 0.6622

- Transitioning from a basic CNN to RESNET50 significantly enhanced accuracy and validation metrics. Deeper architectures excel in feature extraction and representation learning.
- EfficientNetV2L, especially with deeper layer unfreezing, demonstrated robust performance gains. Its efficiency and scalability effectively capture intricate features crucial for emotion recognition.
- Regularization and Stability
- Techniques like dropout, batch normalization, and fine-tuning strategies mitigate overfitting and ensure model stability across architectures.