SWINBURNE VIETNAM

HO CHI MINH CAMPUS

CLASS: **COS30082 – Applied Machine Learning**

Assignment Report

Topic:

Bird Species Classification

FACILITATOR: **Dr. Minh Hoang**

STUDENT NAME:

1. Mai Ngoc Anh Doan - 104220776

TUTORIAL CLASS: Wednesday 1:00PM

HO CHI MINH CITY – OCTOBER, 2024

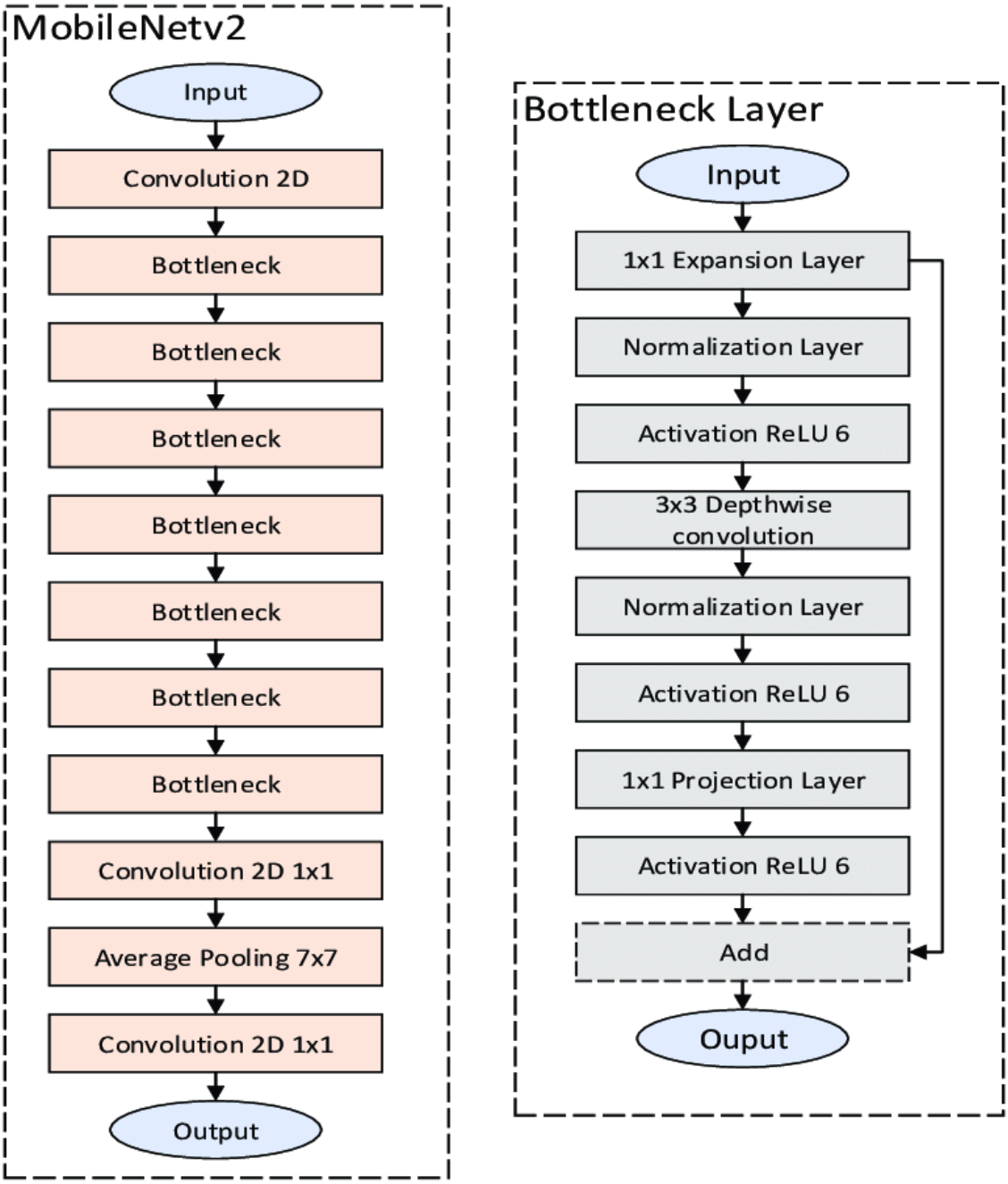
Methodology:

1. Model Architecture:

In this bird species classification assignment, I used two familiar models: MobileNetV2 and ResNet50V2. The reason behind this choice is solely because my laptop is only capable of running these two models, with other models, my computer would decline to train as a result of insufficient memory.

- **MobileNetV2:** This model is known for their efficiency while having a lightweight design. In this assignment, the number of classes is kind of large leading to many models crashing my VSCode by running out of memory. On the other hand, with MobileNetV2 with its depthwise separable convolutions, the parameters are reduced significantly resulting in faster training with high accuracy.

```
Model: "mobilenetv2_1.00_224"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0         []

 Conv1 (Conv2D)              (None, 112, 112, 32)      864       ['input_1[0][0]']

 bn_Conv1 (BatchNormalizati  (None, 112, 112, 32)      128       ['Conv1[0][0]']
 on)

 Conv1_relu (ReLU)           (None, 112, 112, 32)      0         ['bn_Conv1[0][0]']

 expanded_conv_depthwise (D  (None, 112, 112, 32)      288       ['Conv1_relu[0][0]']
 epthwiseConv2D)

 expanded_conv_depthwise_BN  (None, 112, 112, 32)      128       ['expanded_conv_depthwise[0][0
  (BatchNormalization)                                           ]']

 expanded_conv_depthwise_re  (None, 112, 112, 32)      0         ['expanded_conv_depthwise_BN[0
 lu (ReLU)                                                       ][0]']

 expanded_conv_project (Con  (None, 112, 112, 16)      512       ['expanded_conv_depthwise_relu
 v2D)                                                            [0][0]']

...
Total params: 2257984 (8.61 MB)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 2257984 (8.61 MB)
__________________________________________________________________________________________________
```

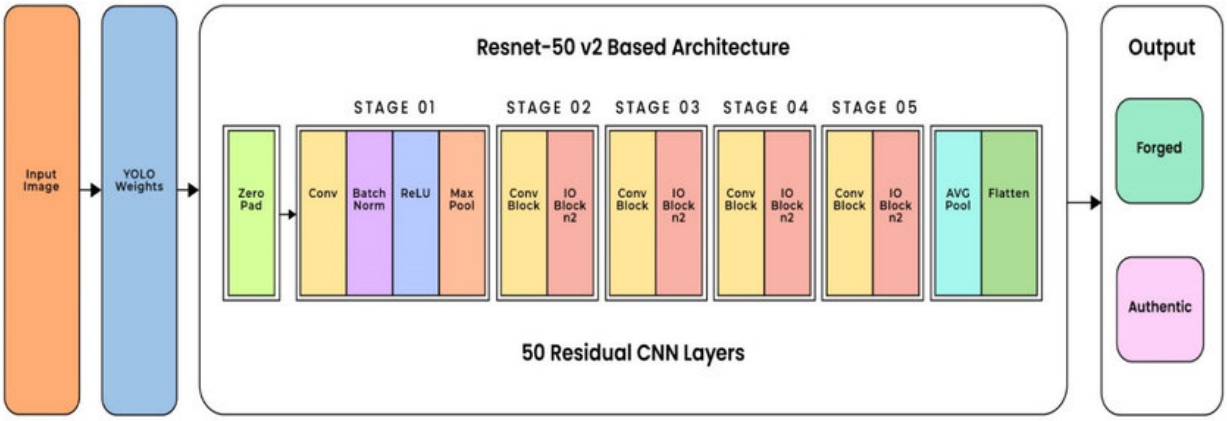


That being said, this architecture makes use of combining bottleneck layers with ReLU6 as an activation function. The last convolution layer would output 200 classes, according to 200 bird species.

- **ResNet50V2:** In contrast to MobileNetV2, this model is chosen for its deeper network design, which can help capture more complex features in the database. Moreover, the

technique used in ResNet50V2 is the residual connection, allowing the model to learn features without encountering vanishing gradient issues.

```
Model: "resnet50v2"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0         []

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)       0         ['input_1[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64)      9472      ['conv1_pad[0][0]']

 pool1_pad (ZeroPadding2D)   (None, 114, 114, 64)      0         ['conv1_conv[0][0]']

 pool1_pool (MaxPooling2D)   (None, 56, 56, 64)        0         ['pool1_pad[0][0]']

 conv2_block1_preact_bn (Ba  (None, 56, 56, 64)        256       ['pool1_pool[0][0]']
 tchNormalization)

 conv2_block1_preact_relu (  (None, 56, 56, 64)        0         ['conv2_block1_preact_bn[0][0]
 Activation)                                                     ']

 conv2_block1_1_conv (Conv2  (None, 56, 56, 64)        4096      ['conv2_block1_preact_relu[0][
 D)                                                              0]']

 conv2_block1_1_bn (BatchNo  (None, 56, 56, 64)        256       ['conv2_block1_1_conv[0][0]']
 rmalization)
...
Total params: 23564800 (89.89 MB)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 23564800 (89.89 MB)
__________________________________________________________________________________________________
```



Like the name, this model has 50 Residual CNN Layers across 5 stages. The special technique used here is skipping connections to keep information from previous layers. This can be beneficial when comes to distinguishing birds' features with similar traits.

2. Data Preprocessing:

- **Data Augmentation:** Different techniques are used not only to ensure that overfitting can not take place but also to contribute to training datasets with many environmental conditions and perspectives of birds' images. These included RandomFlip, RandomRotation, RandomWidth, RandomHeight, RandomZoom, and RandomContrast.

```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.2),
    tf.keras.layers.RandomWidth(0.2),
    tf.keras.layers.RandomHeight(0.2),
    tf.keras.layers.RandomZoom(0.2),
    tf.keras.layers.RandomContrast(0.2)
])
```

- **Normalization:** To ensure consistency between input data, all images are resized to 224x224 and normalized to a range between 0 and 1.

```
rescale = tf.keras.layers.Rescaling(1./127.5, offset=-1)
```

3. Lost Function and Optimization:

- **Loss Function:** The loss function that I have used in this project is Sparse Categorical Cross Entropy. This is used for projects with multi-class classification. It calculates loss from the real distribution of classes and predicted ones.
- **Optimizers:** In the early stages of implementation, Adam is used since it is known for adaptive learning rate capability leading to faster convergence. After reaching the fine-tuning step, the optimizer is changed to RMSProp. Although Adam is superior when comes to optimizers in the later stages of training (fine-tuning phase), RMSProp takes place to handle non-stationary objectives, which usually appear in this phase.

```
base_learning_rate = 0.001
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=[tf.keras.metrics.SparseCategoricalAccuracy(name='accuracy')])
```

```
model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              optimizer=tf.keras.optimizers.RMSprop(learning_rate=base_learning_rate / 100),
              metrics=[tf.keras.metrics.SparseCategoricalAccuracy(name='accuracy')])
```

4. Hyperparameters:

- **Batch Size:** My batch size in this assignment is 32 for both models. At first thought, I wanted to set it larger to prioritize computational speed but since batch size has a big impact on accuracy, I chose 32 to balance between computational speed and memory efficiency.

```
BATCH_SIZE = 32
IMG_SIZE = 224
```

- **Epochs:** Both models would be trained with 20 epochs and 15 fine-tuned epochs. To be honest, I would prefer training more but my laptop can not handle these at all and refuse to work.

```
initial_epochs = 20
```

```
fine_tune_epochs = 15
total_epochs =  initial_epochs + fine_tune_epochs
```

5. Model Training:

Training Process: The training dataset was split into training and validation datasets, with 80% accounted for by the training dataset and 20% images used to validate. In the training phase, the performance of the model is recorded by tracking loss, validation loss, accuracy, and validation accuracy.

```
Epoch 1/20
121/121 [==============================] - 242s 2s/step - loss: 4.5224 - accuracy: 0.1046 - val_loss: 2.6903 - val_accuracy: 0.4124
Epoch 2/20
121/121 [==============================] - 354s 3s/step - loss: 2.7643 - accuracy: 0.3473 - val_loss: 1.8399 - val_accuracy: 0.5430
Epoch 3/20
121/121 [==============================] - 527s 4s/step - loss: 2.0998 - accuracy: 0.4770 - val_loss: 1.3059 - val_accuracy: 0.6477
```

6. Evaluation Metrics:

According to assignment requirements, there are 2 evaluation metrics were used to know whether the model working or not:

- **Top-1 Accuracy:** is employed to evaluate the overall classification performance of the models. This measures how the top predicted class got the actual labels.

$$\text{Top-1 accuracy} = \frac{1}{N}\sum_{k=1}^{N} 1\{argmax(y) == groudtruth\}$$

- **Average Accuracy Per Class:** This is employed to evaluate the model's performance across each individual class, making sure that there is no bias across 200 bird species.

$$\text{Ave} = \frac{1}{C}\sum_{i=1}^{C} T_i$$

7. Reasons for not using CNN:

CNN is not used in this project because it is too simple to handle a big dataset such as CUB-200-2011. Due to its shallow design, the model would be overfitting since it has inadequate architecture to capture complex features between 200 bird species. Additionally, models such as MobileNetV2 and ResNet50V2 are pre-trained models that have been trained on the large dataset resulting in better classifying images kind of tasks.

Results and Discussion

MobileNetV2 Results:

The MobileNetV2 achieves the following results after training for 20 epochs and 15 epochs are used for fine-tuning.

- **Training Accuracy:** For the final epoch, the model reached 87.97% in training accuracy.
- **Validation Accuracy:** For the final epoch, the model reached 88.19% in validation accuracy.
- **Validation Loss:** For the final epoch, the model reached 37.99% in validation loss.

After finishing evaluating model, the results on test dataset comes.

- **Top-1 Accuracy:** Running this model on the test data returned 57.56% in Top-1 Accuracy.
- **Average Accuracy Per Class:** Running this model on the test dataset returned 56.91% in Average Accuracy Per Class.

```
Epoch 34/35
121/121 [==============================] - 482s 4s/step - loss: 0.3888 - accuracy: 0.8833 - val_loss: 0.4018 - val_accuracy: 0.8798
Epoch 35/35
121/121 [==============================] - 434s 3s/step - loss: 0.4102 - accuracy: 0.8797 - val_loss: 0.3799 - val_accuracy: 0.8819
```

```
Top-1 Accuracy: 57.56%
Average Accuracy Per Class: 56.91%
```

With these results, we can see that the MobileNetV2 performed well on the training and validation dataset but had some issues when running the test dataset. In other words, this model found difficulties in handling completely unseen data. Although of this issue, this model looks promising. If more resources and further fine-tuning, the results would be better.

ResNet50V2 Results

The results of ResNet50V2 model after 20 epochs for training and 15 epochs for fine-tuning are listed below:

- **Training Accuracy:** For the final epoch, the model reached 89.16% in training accuracy.
- **Validation Accuracy:** For the final epoch, the model reached 94.61% in validation accuracy.
- **Validation Loss:** For the final epoch, the model reached 17.89% in validation loss.
- **Top-1 Accuracy:** Running this model on the test dataset returned 56.06% in Top-1 Accuracy.
- **Average Accuracy Per Class:** Running this model on the test dataset returned 55.11% in Average Accuracy Per Class.

```
...
Epoch 34/35
121/121 ━━━━━━━━━━━━━━━━━━━━ 254s 2s/step - accuracy: 0.8942 - loss: 0.3742 - val_accuracy: 0.9451 - val_loss: 0.1922
Epoch 35/35
121/121 ━━━━━━━━━━━━━━━━━━━━ 257s 2s/step - accuracy: 0.8916 - loss: 0.3816 - val_accuracy: 0.9461 - val_loss: 0.1789
```

```
Top-1 Accuracy: 56.06%
Average Accuracy Per Class: 55.11%
```

Despite high training and validation accuracy, the results on the test dataset are kind of disappointing to me. For this reason, this model is believed that have been overfitted. This problem can be addressed if there are more computational resources available.

Comparison and Discussion:

Both the MobileNetV2 and ResNest50V2 are trained and fine-tuned with the same epochs but each model reflected a different perspective on strengths and weaknesses following the bird classification task. This will be disclosed in detail below:

Training and Validation Performance:

- MobileNetV2 reached 87.97% in training accuracy and 88.19% in validation accuracy, this is a incredible numbers in machine learning but much smaller than ResNest50V2 model's results with 89.16% in training accuracy and 94.61% in validation accuracy.
- About validation loss, MobileNetV2 had 37.99%, while ResNest50V2 only accounted for 17.89%. With the statistics, it is revealed that ResNest50V2 works excellent in predicting during training and validation phase since the validation is smaller and higher validation accuracy.

Test Data Performance:

With the outstanding statistics during the training and validation phase, both models still experienced a significant drop in test accuracy metrics. The most surprising thing is that the MobileNetV2 slightly outperformed ResNest50V2.

MobileNetV2 is believed to perform better on unseen test data. Although ResNest50V2 outperforms in the validation phase, this evidence shows that the ResNest50V2 model has overfitted to the training and validation dataset. The MobileNetV2 was also overfitted since both models' performance dropped when evaluating the test dataset.

Model Complexity and Hardware Consideration:

- MobileNetV2 is the model that is known for lightweight one, it does not cost so much computational resources to train or fine-tune. Thanks for it, this model can evaluate the test dataset better although being a smaller architecture.

- On the other hand, ResNest50V2 is a deep and complex model that needs more resources to exploit its performance. I believe that hardware plays a vital role in fine-tuning and evaluating test data. With better hardware, the model would perform a better test performance.

Challenges due to Hardware Limitations:

I am aware that there is still room for improvement in both models but the biggest problem that I have faced is hardware's limitation. My laptop can not handle the large workload, especially training deep learning models, which results in low test dataset performance. Moreover, not only do I have trouble fine-tuning the model, but also prevents me from trying other pre-trained models without crashing the application. If my hardware is more dynamic, the performance of both models MobileNetV2 and ResNest50V2 could be improved and I can experience other models.

Conclusion:

In conclusion, MobileNetV2 is more suitable for this bird classification task than other models, such as ResNest50V2 in my project. In contrast, ResNest50V2 also proved that it can be improved with more fine-tuning phases since it suffered from overfitting because of its complexity. Also, a challenge that I am facing is lacking computational resources to optimize both models or experimenting other pre-trained model. In my opinion, I can make this model better by configuring the hyperparameters of the models. Notwithstanding, due to my limitation of hardware, I can not properly configure a model without the application crashing.