Model Optimization and Tuning Phase Template

Date	15 March 2024
Team ID	Team-738168
Project Title	Cognitive Care: Early Intervention for Alzheimer's Disease
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters

DenseNet169	`# Model Initialization`
	
	`base_model = DenseNet169(input_shape=(224,224,3),`
	`                         include_top=False,`
	`                         weights="imagenet")`
	
	base_model: This assigns the created model to a variable named base_model.
	input_shape=(224, 224, 3): This argument defines the expected shape of the input images for the model.
	include_top=False: This argument controls whether to include the pre-trained top layers (classification layers) of the DenseNet169 model.
	weights="imagenet": This argument specifies the pre-trained weights to load into the model.

```
OPT     = tensorflow.keras.optimizers.Adam(lr=0.001)

model.compile(loss='categorical_crossentropy',
              metrics=[tensorflow.keras.metrics.AUC(name = 'auc')],
              optimizer=OPT)
```

OPT = tensorflow.keras.optimizers.Adam(lr=0.001): This line creates an optimizer object named OPT
model.compile(loss='categorical_crossentropy', : This line begins the compilation process for the model you've created.
loss='categorical_crossentropy': This argument specifies the loss function used to measure the model's performance during training.
metrics=[tensorflow.keras.metrics.AUC(name = 'auc')],: This argument defines the metrics you want to monitor during training.
AUC(name='auc'): This calculates the Area Under the Curve (AUC) for a receiver operating characteristic (ROC) curve.
optimizer=OPT): This argument specifies the optimizer you want to use for training the model.

```
model_history=model.fit(train_dataset,
                        validation_data=valid_dataset,
                        epochs = 10,
                        callbacks = callback_list,
                        verbose = 1)
```

model_history : This line assigns the result of the model.fit function to a variable named model_history.
model.fit(...): This is the core function for training the model.

- model: This refers to the Keras model you've already created and defined.

train_dataset: This argument specifies the training dataset you want to use to train the model.
validation_data: This argument (optional) allows you to specify a separate validation dataset.
epochs : This argument defines the number of times the entire training dataset will be passed through the model for training.
callbacks : This argument allows you to specify a list of callback functions that will be invoked during the training process.
verbose : This argument controls the verbosity of the training

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
DenseNet169	• DenseNet169 can be a good candidate model for initial exploration due to its strong feature extraction capabilities. • DenseNet169 has a complex architecture with many layers, potentially leading to overfitting, especially with limited medical datasets. • DenseNet's architecture promotes feature reuse and alleviates the vanishing gradient problem, potentially leading to robust feature extraction from MRI images. • Pre-trained DenseNet169 on large image datasets can be fine-tuned for AD prediction, leveraging its ability to learn generalizable image features. • DenseNet based multi-class categorization network optimized with a focused loss to assess the clinical stage of the predicted brain. • The final DenseNet169 algorithm class categorization of AD diagnosis includes mild demented, moderate demented, non-demented, and very mild demented. As a result, the unified framework allows for the simultaneous optimization of classifierModeling and pattern recognition (from communal space to labeled space and from original feature space to common space, respectively)