Model Optimization and Tuning Phase Template

Date	15 March 2024
Team ID	739868
Project Title	Real Time Communication System Powered By AI For Specially Abled
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

CNN	• **Batch Size**: The number of samples processed before updating the model's weights. Larger batch sizes accelerate training but require more memory, while smaller batches are more memory-efficient.
	  ○ **Example**: batch_size=32
	• **Epochs**: The number of times the entire training dataset is passed through the model. More epochs improve learning but may lead to overfitting.
	  ○ **Example**: epochs=10
	• **Learning Rate**: Determines the step size during optimization. A smaller rate slows down the training process, while a larger rate might cause the model to miss optimal solutions.
	  ○ **Example**: learning_rate=0.001
	• **Optimizer**: Determines how the model's weights are updated based on the loss function. Common optimizers include **Adam** (adaptive learning rate) and **SGD** (stochastic gradient descent).
	  ○ **Example**: optimizer=Adam()
	• **Loss Function**: Measures how well the model's predictions match the true values. **Categorical cross-entropy** is widely used for multi-class classification.

	○ **Example**: loss='categorical_crossentropy'
	
	Initialize the model
	`model = Sequential()`
	
	Add the convolution layer
	`model.add(Convolution2D(32,(3,3),input_shape=(64,64,1),activation = 'relu'))`
	
	Add the pooling layer
	`model.add(MaxPooling2D(pool_size=(2,2)))`
	
	Add the flatten layer
	`model.add(Flatten())`
	
	Adding the dense layers
	`model.add(Dense(units=512,activation='relu'))`
	`model.add(Dense(units=9,activation='softmax'))`

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
CNN	The model processes images, extracts features using convolutional layers, and makes predictions through fully connected layers. It utilizes **ReLU** activation for hidden layers and **Softmax** for multi-class classification. The model is trained with **categorical cross-entropy** loss and the **Adam optimizer**. It's designed to recognize specific patterns or gestures in images, enabling real-time communication for specially-abled individuals.