1. Implemented Improvements and Rationale

The original ViT implementation was enhanced with two major changes: the adoption of the standard **[CLASS] token** mechanism and the addition of a **Learning Rate Scheduler**.

A. Architectural Modification: Implementing the [CLASS] Token

Original Mechanism	Modification	Rationale for Improvement
Feature Aggregation: Flattening all final patch embeddings into a single large vector.	A learnable **[CLASS] token** is prepended to the sequence of patch embeddings before they enter the Transformer blocks. Only the output of this single token is used for final classification.	This is the standard approach from the original ViT paper. It is superior because the [CLASS] token is actively forced, via the **self-attention** mechanism across all layers, to **aggregate global information** from all patches. This results in a more efficient and powerful single vector representation for the classifier head compared to simply concatenating (flattening) all patch outputs.
Code Change:	PatchEncoder was modified to include a new learnable cls_token weight and to adjust the position_embedding dimension by +1. The create_vit_classifier function was changed to select only the first token's output instead of flattening.	

B. Training Mechanism: Adding a Learning Rate Scheduler

Original Mechanism	Modification	Rationale for Improvement
Optimization: Fixed learning rate 0.001 throughout training.	Added the **keras.callbacks.ReduceLROnPlateau** scheduler, monitoring val_accuracy. The learning rate is reduced by a factor that stops improving for a defined patience period.	Deep models like ViTs are sensitive to learning rates. A fixed rate often prevents the model from settling into a sharp optimum later in training, causing loss to oscillate. The scheduler allows for **large initial steps** (fast learning) and then adapts to take **smaller, more cautious steps** as convergence nears, leading to better final model accuracy and stability.
Code Change:	The run_experiment function was updated to include the ReduceLROnPlateau callback in the model.fit call.	

2. Experimental Setup and Hyperparameters

To facilitate comprehensive experimentation (as required by Task 3), the codebase was modularized to easily switch between datasets and modify core model configurations.

A. Flexible Data and Model Configuration

- A new function, load_datadataset_name was introduced to load either **CIFAR-10** or **CIFAR-100** by setting the global dataset_name variable.
- The create_vit_classifier function was updated to accept transformer_layers (Depth) and projection_dim (Width) as arguments, enabling quick iteration over different model architectures.

B. Base Hyperparameters Used in Experiments

The following base configuration was used for the Improved Model setup:

Parameter	Value	Description
Learning Rate	0.001	Base learning rate for the AdamW optimizer.
Weight Decay	0.0001	Regularization to prevent overfitting.
Batch Size	256	Standard batch size.
Image Size	72 x 72	Input image resize dimension.
Patch Size	6 x 6	Size of input patches, resulting in 144 patches.
Projection Dim (Width)	64	Dimension of patch embeddings.
Transformer Layers (Depth)	8	Number of attention blocks.
LR Scheduler	ReduceLROnPlateau	Factor 0.5, Patience 5, min_lr 1e-6