Week 4 Assignment: CNNs Vs. Transformers

Candidate: Sneha Santha Prabakar

Coding platform: Google Collab

Please Note:

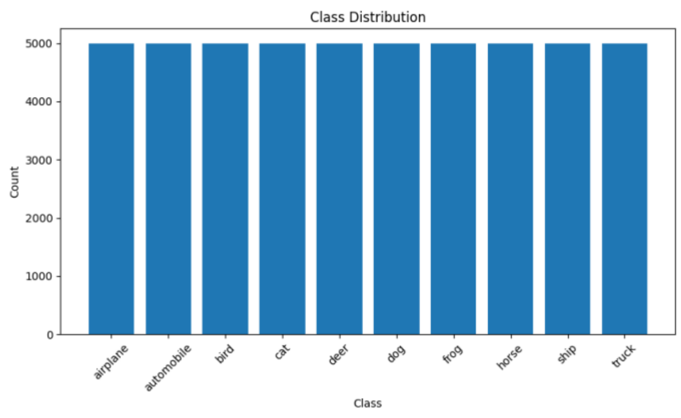
The python notebook included in the submission contains all the charts and steps mentioned in the grading criteria. This PDF document only provides a summary for each section executed in the python notebook. Hence, please refer to the python notebook for the execution steps.

Part 1: Data summary

- Total data size: 60,000 rows
- Pixels: 32 x 32
- Each image has a height and width of 32 pixels
    - This suggests that these are small, low-resolution images
- Colour channels: 3
    - Each image has 3 colour channels - RGB (red, green, blue)
- The training set contains 50,000 images The test set contains 10,000 images
- There are 10 labels for the data:
    - Animals: bird, cat, deer, dog, frog, horse
    - Automobiles: airplane, automobile, ship, truck
- Normalization: **Division by 255** as this is an image data with pixel values typically in the range [0, 255] representing 8-bit color channels. Neural networks train more efficiently when inputs are on a standardized scale—ideally in the [0, 1] range or zero-centered like [-1, 1]. Here, we normalize the pixel values to be in the **[0,1] range**.

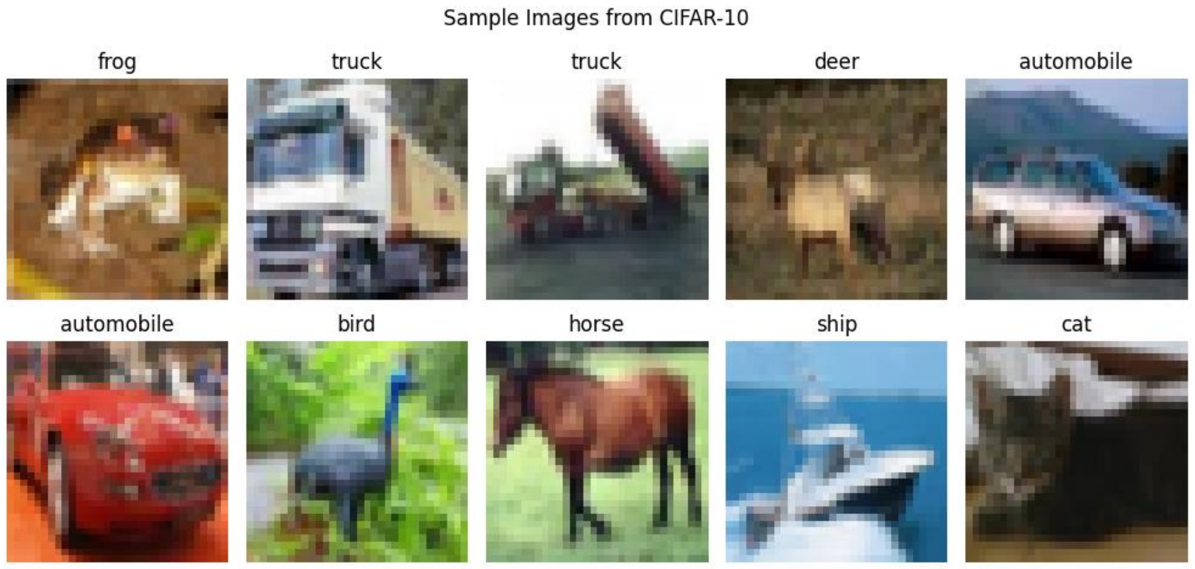
Part 2: Data Pre-Processing

Class Distribution



The classes are uniformly distributed with equal number of samples for each class.

Sample Image Analysis



From visual analysis of the sample images, we can make the following observations:

- Diversity: The CIFAR-10 dataset includes a variety of object types, such as animals (e.g., frog, deer, bird), vehicles (truck, automobile, ship), and scenes that differ significantly in texture and background complexity. Most of the images have complex, real-world backgrounds, which require the model to effectively separate foreground objects from context noise.

- Low resolution: Images are small (32x32 pixels), leading to visible blurring or pixelation. This may make it more difficult for the model to capture fine-grained details, such as distinguishing between a truck and an automobile.

- Centered objects: In most images, the objects are in the center of the frame, which can make classification easier for CNNs.

Part 3: CNN model

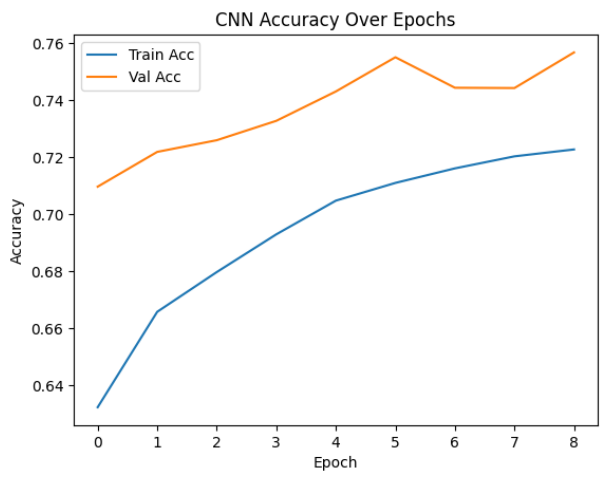
Constructing the model:

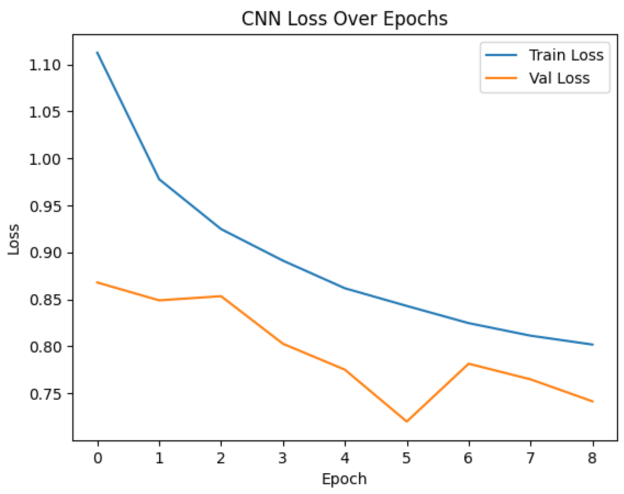
- We start with 3 convolutional layers to construct a moderately deep CNN as our data size is relatively small.
- We will also introduce Pooling after each convolutional block to reduce overfitting, improve efficiency, and distil key features.
- We will have just one hidden dense layer as it is sufficient when combined with convolutional layers. It balances model complexity vs. training time
- Optimizer function: Adam
- We will also use Dropout and Early stopping for regularisation.

Step-by-step explanation:
- Layer: Conv2d(32 filters)
    - Applies 32 filters of size 3x3 over the image to extract low-level features like edges, textures, and colors.
    - Padding = same spatial dimensions (32×32), which is useful for stacking layers without shrinking the image too quickly.
    - Activation function: ReLU (to introduce non-linearity)
- Layer: MaxPooling2D

  - Reduces the spatial size (height and width) by taking the maximum value in each 2x2 patch.
  - This Introduces spatial invariance, making the model more robust to shifts and distortions and it also prevents overfitting by reducing the feature map size.
- Layer: Conv2D(64 filters) and MaxPooling2D
  - Deeper layers learn more complex features (like shapes, object parts).
  - Increasing filters from 32 → 64 gives the network greater capacity to learn richer representations.
  - Two pooling layers help aggressively reduce spatial dimensions (from 32x32 to 8x8), which is important for final classification while retaining useful features.
- Layer: Flatten
  - Flattens the 3D feature maps (from previous layers) into a 1D vector so it can be fed into dense (fully connected) layers.
- Layer: Dense(64)
  - A fully connected layer with 64 neurons that combines all the extracted features to make sense of the patterns.
  - 64 units is a balanced choice as it is small enough to avoid overfitting, large enough to allow meaningful combinations.
- Layer: Dropout
  - Randomly drops 30% of the neurons during training.
  - Helps prevent overfitting.
- Layer: Dense(10)
  - Output layer with 10 neurons — one for each class in CIFAR-10.
  - Activation function: Softmax (to convert logits to probabilities, for the multiclass classification)

Training and Loss Curve:





Training accuracy steadily improves from ~63% to ~72% over 9 epochs. **Validation accuracy** consistently stays higher, peaking around ~75.8%. This suggests that the model is learning effectively and there is no overfitting.

Training loss decreases steadily, from ~1.1 to ~0.80. **Validation loss** also decreases and stays **lower than training loss**. Some mild fluctuation is visible (typical with real-world data), but no sharp increases.

The CNN model shows healthy convergence, with both training and validation accuracy improving steadily. Validation accuracy exceeds training accuracy slightly, suggesting effective generalization likely aided by data augmentation and regularization. The loss curve supports this observation, with both training and validation losses declining consistently across epochs.

Part 4: ViT

Constructing the model:

Patch Extraction: Images of size 32×32 are divided into smaller patches of size 4×4, resulting in 64 patches per image.

Patch Encoding: Each patch is linearly projected into a feature vector and enriched with positional embeddings to retain spatial information.

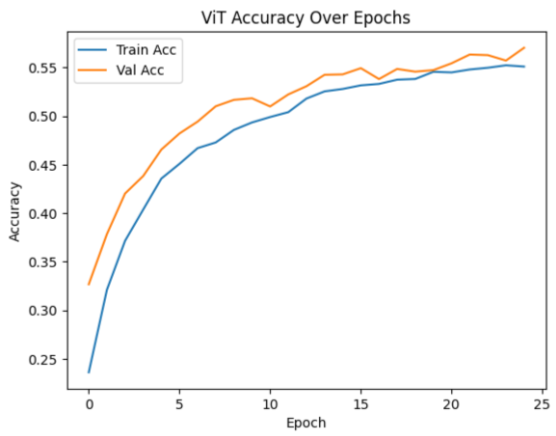
Transformer Encoder: The encoded patches are passed through a stack of 4 transformer blocks, each consisting of:

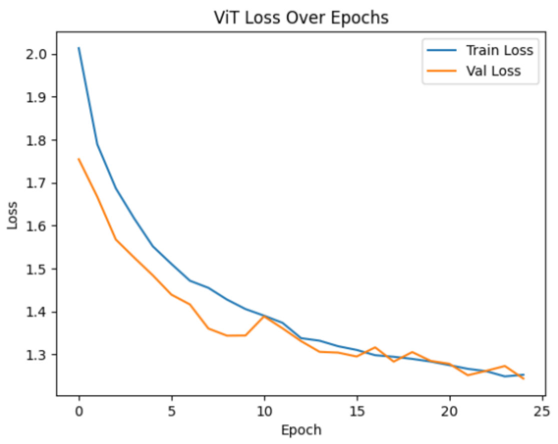
- Multi-Head Self Attention (2 heads)
- Layer Normalization
- Feedforward Neural Network (MLP)
- Residual Connections

Key training decisions:

- Optimizer: AdamW (Adam with weight decay)
- Loss Function: Categorical crossentropy (with one-hot encoded labels)
- Learning Rate Schedule: ReduceLROnPlateau
- Regularization:
  - Dropout (0.2 in transformer and classifier layers)
  - EarlyStopping (to prevent unnecessary training after convergence)
  - Weight Decay (L2 regularization via AdamW)

Training and Loss Curve:





Accuracy Curve:

Steady improvement in both training and validation accuracy.

- Final training accuracy: ~55%
- Final validation accuracy: ~55.5%

The closeness of the two curves suggests the model is not overfitting.

Loss Curve:

- Both training and validation loss decrease smoothly and converge around 1.27–1.29.
- No divergence or sharp increase in validation loss, indicating stable generalization.

Part 5: Comparison of CNN vs. ViT

Performace Gap

- Final Train Accuracy: CNN (72.3%) >> ViT (55.2%)
- Final Validation Accuracy: CNN (75.8%) >> ViT (55.5%)
- Final Validation Loss: CNN (0.73) << ViT (1.28)

The CNN model outperformed the ViT by over 20 percentage points in validation accuracy.

CNNs are better suited for small datasets like CIFAR-10 due to their inductive bias (e.g., local spatial filtering), which allows them to generalize faster with fewer samples.

Training Efficiency

- Avg. Time per Epoch: CNN (6-8 seconds) << ViT (20-22 seconds)
- Convergence Rate: Much faster for CNN than ViT. CNN achieved good accuracy in less than 10 epochs, whereas ViT accuracy plateaued near epoch 20.

CNNs are more computationally efficient and reach higher accuracy faster.

ViTs took longer per epoch and more epochs to plateau — typical for transformer-based models without pretraining.

Learning Pattern

- Training vs. Validation accuracy:
  - CNN: Validation slightly higher - no overfit
  - ViT: Very close - well regularized
- Loss curve:
  - CNN: Smooth, sharp drop, stable
  - ViT: Smooth but plateaued early
- Generalisation ability:
  - CNN: Strong
  - ViT: Limited on this dataset due to no inductive bias or pretraining

CNN learned local features more quickly, helping it to generalise well. ViT lacked inductive biases, requiring more data to perform competitively. Its accuracy plateaued early, suggesting it learned some meaningful representations, but not enough to compete with CNN.

Final conclusion

While the ViT model was correctly implemented and trained with attention mechanisms, regularization, and learning rate scheduling, it was outperformed by the CNN in nearly all aspects due to the nature of the dataset. This outcome aligns with known behavior of ViTs: they perform best with large datasets or pretrained weights, while CNNs are highly effective even on small datasets like CIFAR-10.