Describe my Neural Network Architecture

Intermediate Blocks (`IntermediateBlock`):
- Convolutional layers followed by a fully connected layer.
- The convolutional layers employ learnt filters to capture various features in the input image.
- A fully linked layer then generates a set of weights for these features.
- Another 'IntermediateBlock' produces a weighted combination of its convolutional layer outputs, resulting in an improved feature map for the following layers.

Output Block (`OutputBlock`)
- This is my network's final block.
- It gets processed picture data from the last 'IntermediateBlock'.
- The block averages the features from all channels and routes them through one or more fully connected layers.
- The result is a logit vector containing the network's predictions for each class.

- **Custom Neural Network (`CustomNet`)**:
- We define a custom neural network called 'CustomNet' that includes the intermediate and output blocks.
- 'CustomNet' starts with an 'IntermediateBlock' that accepts raw picture input. After processing via this block, the data is sent through a pooling layer to minimise dimensionality.
- The output is then fed into another 'IntermediateBlock', followed by the 'OutputBlock', which produces the final classification.

Hyperparameters and techniques employed for training.

Configure the Loss Function and Optimiser:
- **Loss Function**: We apply cross-entropy loss, widely used in multi-class classification issues. This loss function is well-suited for our work because it analyses the performance of a classification model that outputs a probability value between 0 and 1
- **Optimiser**: Stochastic Gradient Descent (SGD) was selected as the optimiser for the basic architecture stage of our project. It is a simple but powerful optimisation algorithm. We will experiment with different optimisers, such as Adam and AdamW during when we try to improve our results.

Training Process: The network is trained across many epochs, each representing a complete trip across the training dataset. In each epoch, we loop through the training dataset in batches, as specified by our Data Loader. For each batch, we follow the instructions below:
- **Forward Pass**: The model makes predictions based on the current values of its parameters (weights and biases).
- **Loss Calculation**: We estimate the loss by comparing the model's predictions to the actual labels.
- **Backward Pass**: Using the loss, we execute backpropagation to compute the gradients of the loss concerning each parameter of the model.
- **Parameter Update**: The optimiser adjusts the parameters to reduce the loss.

Data Augmentation: To enhance the model's ability to generalise and to mitigate overfitting, the following data augmentation techniques were applied to the training dataset:
- Random horizontal flipping

- Random cropping with padding
- Random affine transformations
- Random rotations
- Color jittering (adjustments in brightness, contrast, saturation, and hue)

Hyperparameters:

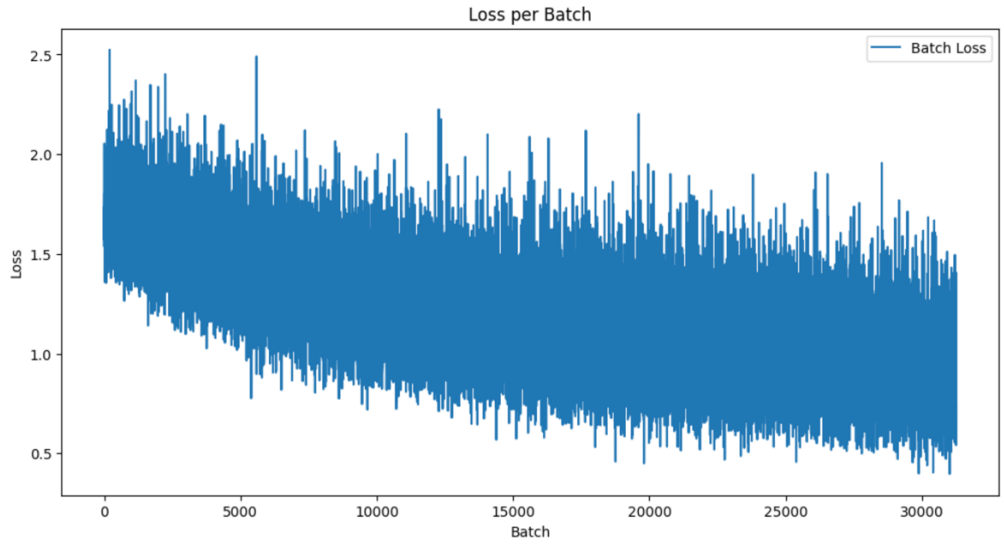
- **Batch Size:** The batch size determines how many samples are propagated across the network simultaneously. Smaller batch sizes frequently produce a regularising impact and reduce generalisation errors.
- **Learning Rate:** This is a crucial hyperparameter that determines the convergence rate of the loss function. It controls the step size used in gradient descent to update the model's weights.
- **Optimisers:**
    - **Stochastic Gradient Descent:** It uses a basic updating rule with a fixed learning rate and is well-known for its reliability.
    - **Adam (Adaptive Moment Estimation):** An advanced optimiser that computes adaptive learning rates for each parameter, resulting in faster convergence.
- **Loss Function:**
    - **Cross-entropy loss:** A commonly used loss function for multi-class classification. It quantifies the difference between the expected and actual probability distributions, providing an effective measure of model performance.
- **Number of Epochs:** The number of epochs indicates how many times the learning algorithm will run through the full training dataset. The optimal number of epochs strikes a balance between underfitting and overfitting.
- **Weight decay:** A regularisation technique that adds a penalty term to the loss function proportionate to the number of weights. It promotes the model to use tiny weights, resulting in simpler models and lowering the danger of overfitting.
- **Dropout Rate:** Determines the likelihood of input items being dropped to prevent overfitting. Dropout is a type of regularisation in which a fixed number of units are removed at random from a network layer during training.

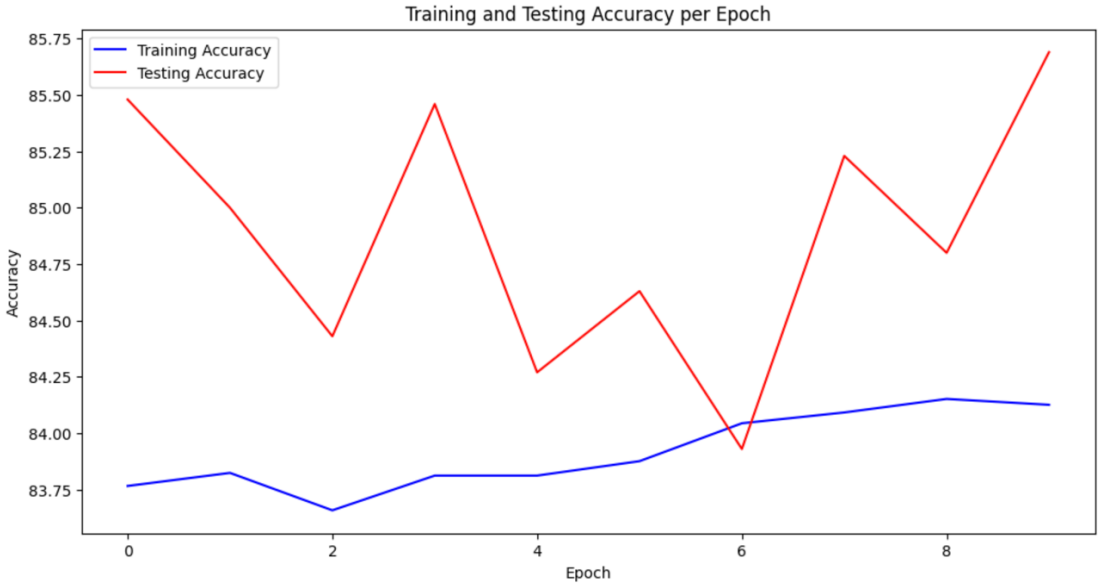
Neural Network that achieved the highest accuracy in the testing dataset

```
Epoch 1: Train Loss: 0.525, Train Acc: 83.77%, Test Acc: 85.48%
Epoch 2: Train Loss: 0.524, Train Acc: 83.82%, Test Acc: 85.00%
Epoch 3: Train Loss: 0.523, Train Acc: 83.66%, Test Acc: 84.43%
Epoch 4: Train Loss: 0.518, Train Acc: 83.81%, Test Acc: 85.46%
Epoch 5: Train Loss: 0.522, Train Acc: 83.81%, Test Acc: 84.27%
Epoch 6: Train Loss: 0.520, Train Acc: 83.88%, Test Acc: 84.63%
Epoch 7: Train Loss: 0.516, Train Acc: 84.04%, Test Acc: 83.93%
Epoch 8: Train Loss: 0.514, Train Acc: 84.09%, Test Acc: 85.23%
Epoch 9: Train Loss: 0.508, Train Acc: 84.15%, Test Acc: 84.80%
Epoch 10: Train Loss: 0.513, Train Acc: 84.13%, Test Acc: 85.69%
Finished Training
```

Highest Test Accuracy achieved is 85.69% as shown above.

Plot of the loss for each training batch, and a plot of the training accuracy





To summarise, designing and enhancing the architecture for image classification on the CIFAR-10 dataset has been a thorough journey of applying practical machine learning techniques. We successfully built a neural network that not only meets the challenge of classifying complex images, but also achieves a commendable testing accuracy of 85.69%, thanks to iterative design enhancements, meticulous hyperparameter tuning, and strategic incorporation of data augmentation techniques.