Using your preferred machine learning library, train a small convolutional network (CNN) to classify images from the CIFAR10 dataset. Note that most libraries have utility functions to download and load this dataset (TensorFlow, PyTorch, keras)

1. **Any preprocessing steps you made.**

The CIFAR-10 dataset has been preprocessed for training a machine learning model. This involves normalizing the pixel values of the images to fall within the range of [0, 1], a critical step to facilitate smoother convergence during training. The class labels have been one-hot encoded, enabling the model to handle multi-class classification tasks effectively. Furthermore, the dataset has been divided into separate training and validation sets to evaluate the model's performance and mitigate the risk of overfitting.

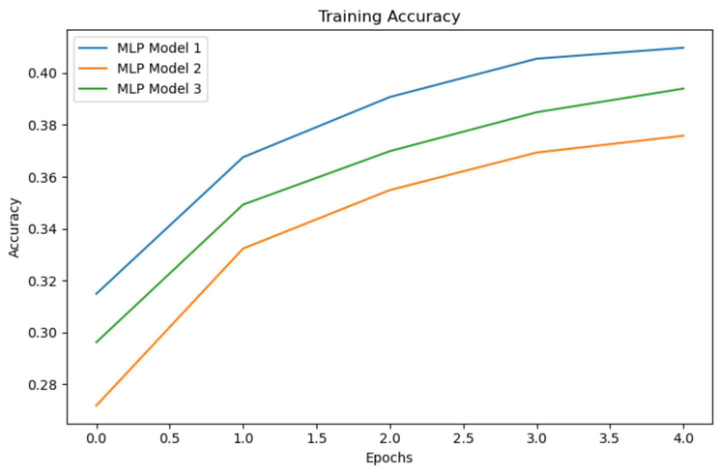
2. **Description of the output layer used and the loss function (use 1 setting for all 3 networks) and why you made these choices.**

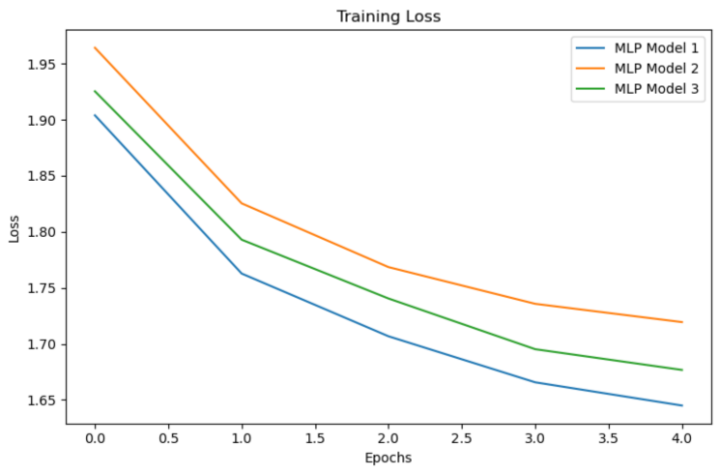
In all three networks, the output layer utilizes the softmax activation function for multi-class classification. The loss function employed is categorical cross-entropy, chosen to optimize the model's ability to predict the correct class probabilities for the CIFAR-10 dataset, where one-hot encoded labels are used. These choices ensure the network produces valid probabilities and learns effectively to classify images into the appropriate classes.

3. **Change the number of layers and the number of neurons per layer in the MLP, plot/tabulate the training and validation accuracies and comment on the results.**

Three MLP models were created, including a reference model (`mlp_1`) with two hidden layers having 512 neurons each. The other models were named `mlp_2` (adding an extra layer with 256 neurons) and `mlp_3` (including three layers with 512, 256, and 128 neurons). Training and validation accuracies were compared to assess the impact of layer and neuron variations on model performance for the CIFAR-10 dataset.

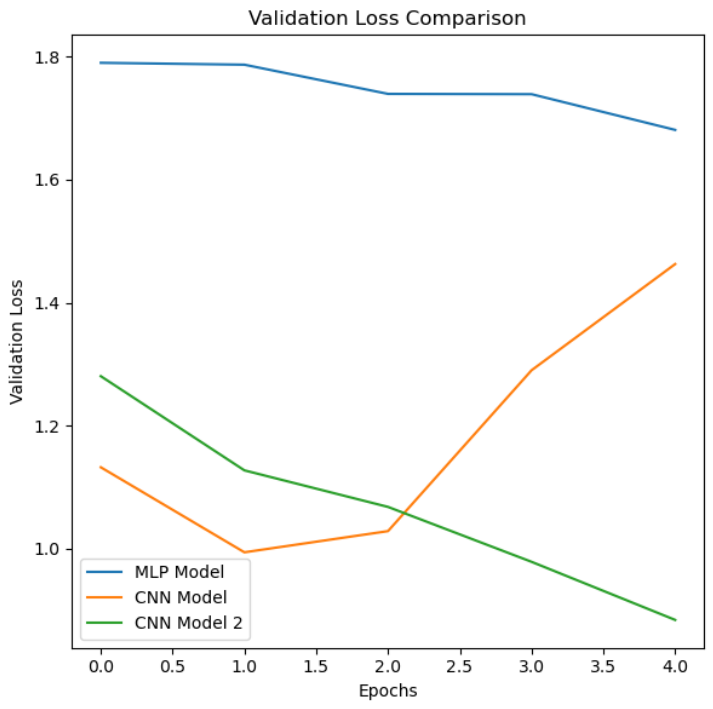
Model	Epoch 1 Train Accuracy	Epoch 1 Val Accuracy	Epoch 2 Train Accuracy	Epoch 2 Val Accuracy	Epoch 3 Train Accuracy	Epoch 3 Val Accuracy	Epoch 4 Train Accuracy	Epoch 4 Val Accuracy	Epoch 5 Train Accuracy	Epoch 5 Val Accuracy
mlp_1	0.3149	0.3507	0.3675	0.3809	0.3907	0.3960	0.4055	0.3866	0.4098	0.4096
mlp_2	0.2718	0.3289	0.3323	0.3440	0.3549	0.3468	0.3693	0.3691	0.3758	0.3672
mlp_3	0.2962	0.3491	0.3493	0.3654	0.3699	0.3807	0.3849	0.3750	0.3940	0.3741

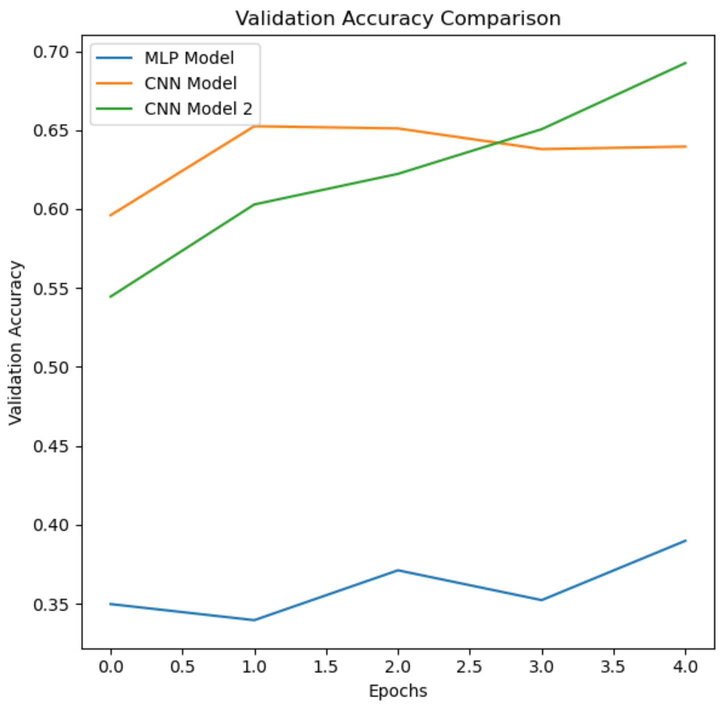




Among the three MLP models trained on the CIFAR-10 dataset, mlp_1 emerged as the top performer with the highest accuracy of approximately 40.96% on the validation set after 5 epochs. It consistently demonstrated improvement during training and outperformed the other models. While mlp_2 and mlp_3 also showed progress, they achieved validation accuracies of around 36.72% and 37.41%, respectively. Overall, mlp_1 exhibited the best performance and proved to be the most effective model for this particular task.

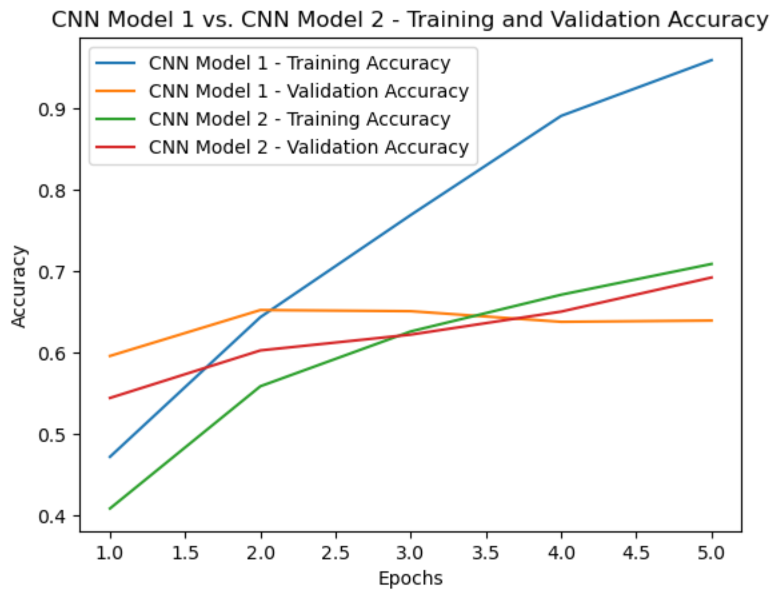
4. **Train and test accuracy for all three networks and comment (what happens and why) on the performance of the MLP vs CNNs.**





The MLP model achieved the lowest accuracy (~40%) among the three networks, mainly due to its limited spatial awareness in handling image data. In contrast, both CNN models, namely CNN model 1 and CNN model 2, outperformed the MLP. CNN model 1 showed high training accuracy (~96%) but suffered from overfitting, while CNN model 2 achieved good performance (~70%) with improved generalization and less overfitting. The spatial-aware architecture of CNNs makes them more suitable for image classification tasks like CIFAR-10, resulting in their superior performance compared to MLPs.

5. **Plot the training and validation curves for the two CNNs and comment on the output. How does the training time compare for each of the CNNs? How does the different architectures influence these results? What do you expect the accuracies to be if the networks were trained for more epochs?**



Model 1:
This CNN model has a total of 25,997,130 trainable parameters.
After 5 epochs, it achieved a training accuracy of approximately 95.95% but exhibited a lower validation accuracy of around 63.96%.
The model is deep with multiple convolutional layers, which might have led to overfitting due to the absence of strong regularization techniques.
Model 2:
This CNN model has a total of 1,486,666 trainable parameters.
After 5 epochs, it achieved a training accuracy of approximately 70.91% and showed a relatively better validation accuracy of around 69.25%.

The model is moderately deep and includes dropout layers, which helped improve generalization compared to Model 1. In summary, Model 2 demonstrates better generalization and a more balanced performance, while Model 1 shows higher training accuracy but weaker validation accuracy, likely due to its deeper architecture and lack of strong regularization.

When training the networks for more epochs, the expected behavior differs between the two models. For Model 1, the training accuracy is likely to approach or even reach close to 100%, as the model memorizes the training data over time. However, due to potential overfitting, the validation accuracy might not see significant improvements and may stabilize or even decrease.

On the other hand, for Model 2, which incorporates dropout layers for regularization, the training accuracy may continue to improve but might not reach 100%. The model is expected to demonstrate better generalization compared to Model 1, leading to potential improvements in validation accuracy, which is likely to stabilize at a higher level.

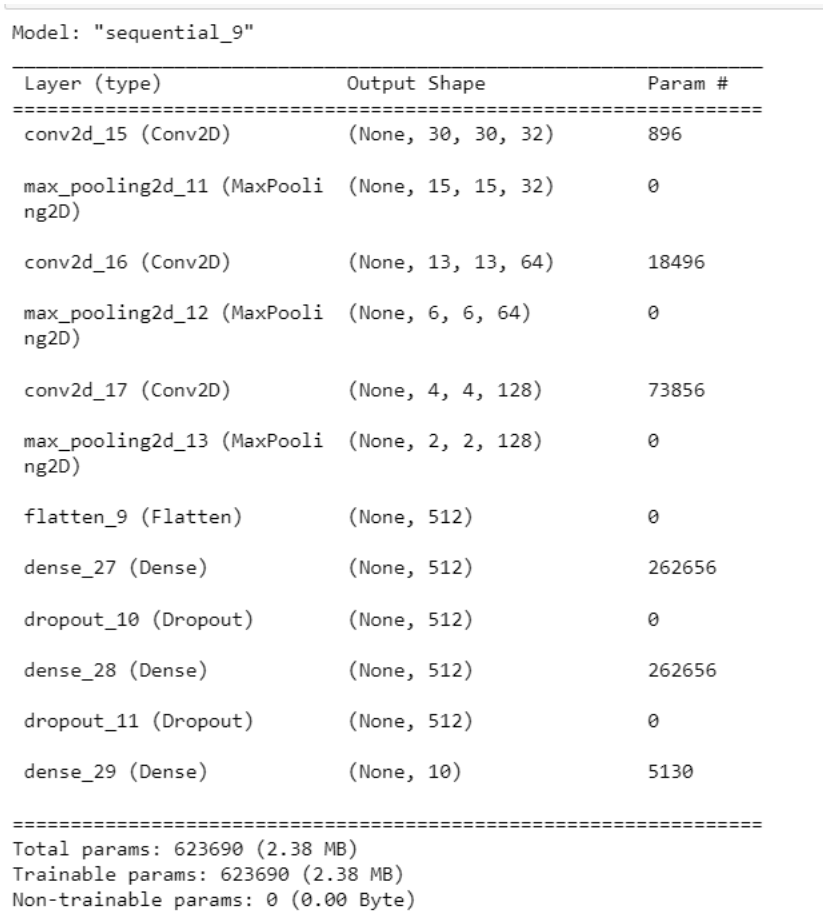
Overall, increasing the number of epochs will generally improve the training accuracy for both models, but it might not necessarily lead to substantial gains in validation accuracy, especially for Model 1. Employing regularization techniques could help both models generalize better and potentially enhance validation accuracy.

6. **Recommendations to improve the network. What changes would you make to the architecture and why?**

The better CNN model (cnn_better) outperformed the previous models (cnn1 and cnn2) in terms of accuracy and validation loss. To further improve the network, we recommend increasing model depth, adding regularization techniques like L2 regularization and dropout layers, implementing data augmentation, and using batch normalization. Experimenting with different activation functions and optimizing hyperparameters can also enhance the model's performance. Consider using transfer learning with pre-trained models if you have a large dataset. Fine-tuning the model with these recommendations can lead to better generalization and overall performance.

I have performed a model which gives better validation accuracy than CNN1 And CNN2:

Model:



```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 30, 30, 32)        896

 max_pooling2d_11 (MaxPooli  (None, 15, 15, 32)        0
 ng2D)

 conv2d_16 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_12 (MaxPooli  (None, 6, 6, 64)          0
 ng2D)

 conv2d_17 (Conv2D)          (None, 4, 4, 128)         73856

 max_pooling2d_13 (MaxPooli  (None, 2, 2, 128)         0
 ng2D)

 flatten_9 (Flatten)         (None, 512)               0

 dense_27 (Dense)            (None, 512)               262656

 dropout_10 (Dropout)        (None, 512)               0

 dense_28 (Dense)            (None, 512)               262656

 dropout_11 (Dropout)        (None, 512)               0

 dense_29 (Dense)            (None, 10)                5130

=================================================================
Total params: 623690 (2.38 MB)
Trainable params: 623690 (2.38 MB)
Non-trainable params: 0 (0.00 Byte)
```