Video link:
https://drive.google.com/file/d/1ckbwUbGBkL4TK897A0vh2IcUKbVQGtq_/view?usp=sharing

Basic ideas for improving the CNN models are as below:

1. Increase the number of epochs: The model may need more training epochs to converge to a better solution. Try increasing the number of epochs in the fit() function.

2. Add batch normalization: Batch normalization can help to improve the stability and convergence of the model. You can add batch normalization layers after each convolutional layer and before the activation function.

3. Use a different activation function: The tanh activation function used in the current model may not be the best choice for this problem. You can try using other activation functions such as ReLU or ELU.

4. Add dropout regularization: Dropout can help to prevent overfitting by randomly dropping out some of the neurons during training. You can add dropout layers after the dense layers.

5. Use data augmentation: Data augmentation can help to increase the size of the training set and improve the generalization of the model. You can use the ImageDataGenerator class in Keras to apply various image transformations to the training data.

LeNet5:

The given model has Accuracy of 0.967 while validation_accuracy is only 0.592. So we made improvements to the baseline model as below to achieve a higher val_acc rate.

1. Added batch normalization layers after each convolutional layer and before the activation function.
2. Added dropout regularization after the first dense layer.
3. Used data augmentation to increase the size of the training set and improve the generalization of the model.

This helped us improve the val_acc to ~0.8.

AlexNet:

In the given code, we have added batch normalization layers, increased the number of filters, used a learning rate scheduler, and applied data augmentation techniques to the training dataset. Additionally, we have used the initialization to initialize the weights of the model.

While the baseline model has acc = 0.6 & val_acc = 0.5, the improved model has acc = 0.85 & val_acc = 0.65

Vgg16:

1. Used the given Vgg16 model and drew the Confusion matrix as well as precision-recall curve.

Precision-Recall curve: This is a 2D curve that shows the trade-off between precision and recall for different threshold values of the classifier. This can be helpful in understanding the balance between true positive rate and false positive rate, and can be especially useful when working with imbalanced datasets.

Vgg19:

For the given Vgg19 model here, we have plotted the precision-recall curve for all classification classes provided in the dataset.