Model

model.add(layers.Conv2D(32, (5, 5), activation='relu', input\_shape=(32, 32, 1)))

model.add(layers.MaxPooling2D((2, 2)))

#model.add(layers.Conv2D(64, (3, 3), activation='relu'))

#model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu', padding = "same"))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(256, (3, 3), activation='relu', padding = "same"))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(500, activation='relu'))

model.add(layers.Dropout(0.5))

model.add(layers.Dense(250, activation='relu'))

model.add(layers.Dropout(0.5))

model.add(layers.Dense(100, activation='softmax'))

# Compile the model

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

on

history = model.fit(self.X\_train, self.y\_train, epochs=25, validation\_data=(self.X\_valid, self.y\_valid), batch\_size=100, verbose=1, shuffle=1)

Result:

Change:

Uncommented out :

#model.add(layers.Conv2D(64, (3, 3), activation='relu'))

#model.add(layers.MaxPooling2D((2, 2)))

Result:



Much worse.

Change:

Removed layer and changed from 256 to 128 and changed from 128 to 64

Result:



Accuracy small bit worse but loss is a lot better

Change:

Changed epochs from 25 to 100

Result:



Loss much higher accuracy pretty much the same

Change:

Removed dropout layer under first dense layer, back to 25 epochs

Results:



Loss much worse and accuracy worse.

Changes:

Added dropout layer back and Increased dropout layer to 0.8 from 0.5 and went back to 50 epochs

Results:



Not much change from previous models

Change:

Added new drop out layer

Result:



Best results by a long shot

Change:

Added another dropout layer

Result:



Best results so far

Change:

Changed dropout layer to 0.4

Result:

Change:

Changed epochs to 150 and batch size=200

Result :



Not much difference [for 150 epochs ]on validation train still low value and loss is high.

Learning rate has been added to Adam optimizer

learning\_rate=0.001

optimizer= Adam(learning\_rate=learning\_rate)

Batch\_size=400 and epochs=50

history = model.fit(self.X\_train, self.y\_train, epochs=50, validation\_data=(self.X\_valid, self.y\_valid), batch\_size=400, verbose=1, shuffle=1)

Also the test\_split is increased to 30%.

self.X\_train, self.X\_valid, self.y\_train, self.y\_valid = train\_test\_split(self.X\_train, self.y\_train, test\_size=0.3, random\_state=42)

Result:



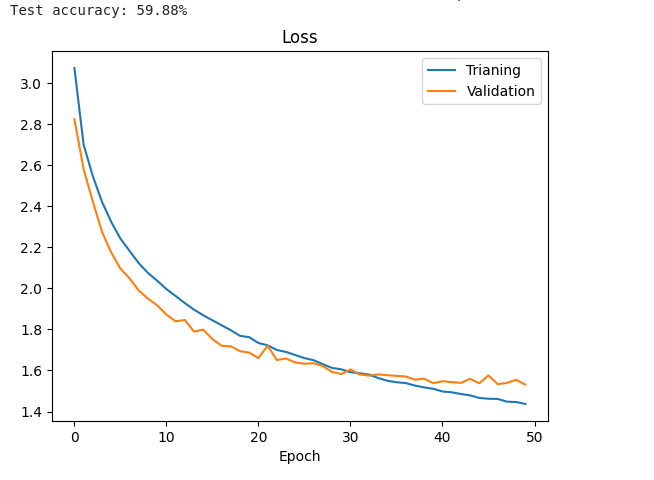
Change:

Learning\_rate=0.01

Validation set= 20%

Batch\_size=300

Epochs same=50

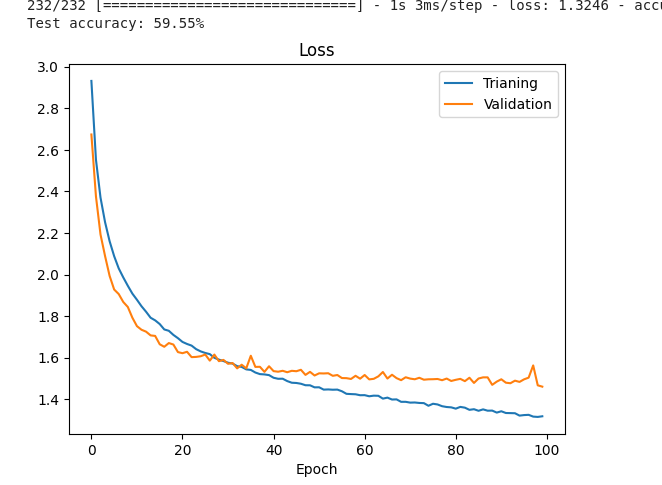
Result

Change :

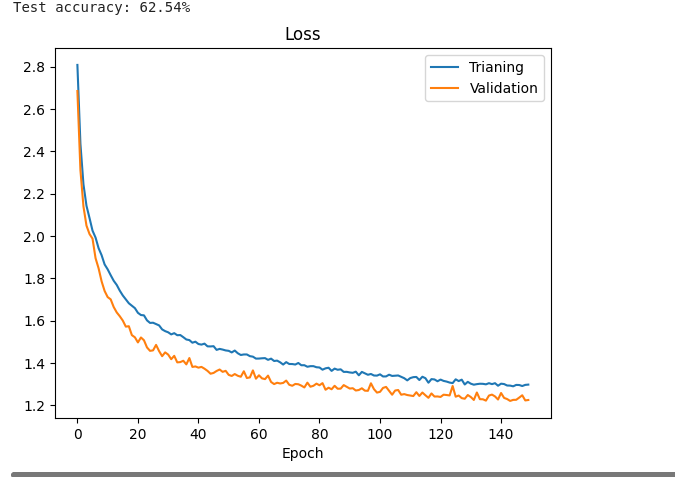
Batch\_size=200 smaller size to prevent overfitting.

Learning\_rate =0.001

Epochs=100

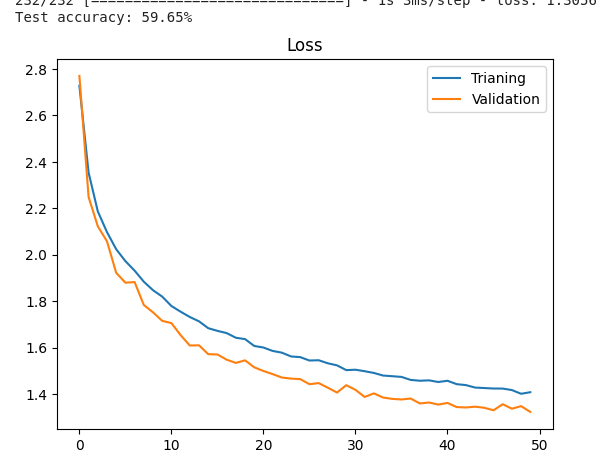
Result :

Change:

Result:

Change :

Learning rate= 0.01

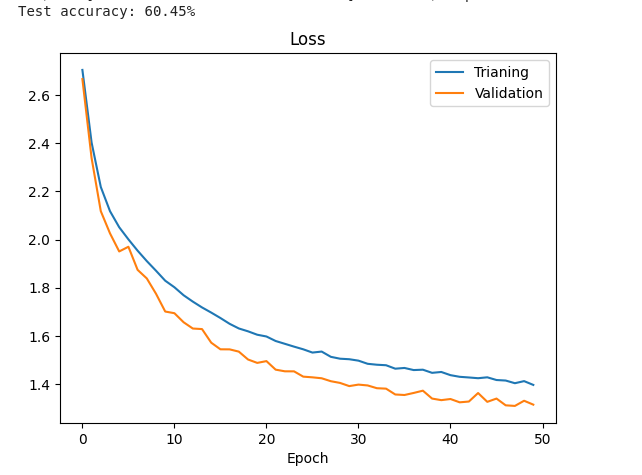


Change:

Epohcs=50

Batch\_size=250

Learning rate=0.01 (It doesn't seem to affect too much either 0.001 or 0.01)



Change:

Adding new layer

Increase dropout

model.add(layers.Conv2D(256, (3, 3), activation='relu', padding="same"))

model.add(layers.MaxPooling2D((2, 2))) -->>new layer added

model.add(layers.Dropout(0.4))-->> new layer added

model.add(layers.Flatten())

model.add(layers.Dense(500, activation='relu'))

model.add(layers.Dropout(0.5)) --->increased from 0.4

model.add(layers.Dense(250, activation='relu'))

model.add(layers.Dropout(0.6)) --->> increased from 0.5

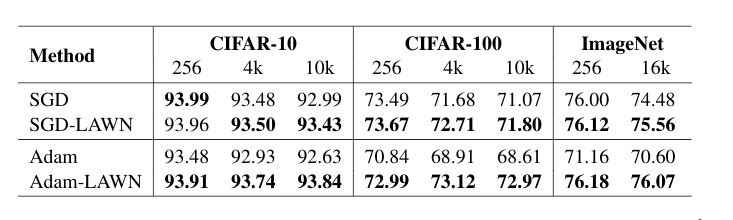
model.add(layers.Dense(100, activation='softmax'))

Result:

NOT GOOD. Test Accuracy is 13%

Observation:

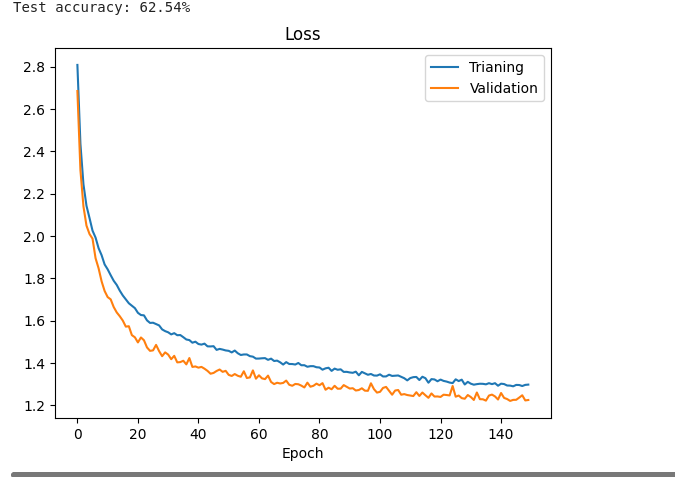
At 40 epochs out of 100 epochs it reaches test accuracy at around 56 % and final value after running all 100 epochs is very close to 40 epochs, which is 59%. It seems that model reaches a plateau.

 (et al Gupta 2021)

From this scientific paper((et al Gupta 2021)) we can see that with Adam optimizer they didn't break 70% for Test Accuracy and batch size 4k or even 10k

FINAL Model.:





References List:

1.Gupta, A., Ramanath, R., Shi, J., & Keerthi, S. S. (n.d.). *OPT2021: 13th Annual Workshop on Optimization for Machine Learning Adam vs. SGD: Closing the generalization gap on image classification*.