**Report:**

Here we consider a subset of data from the data set that we have and divide the data into train, test and validation.

In convolution neural networks we use many layers but here in this case we use a convolution2d which is used to analyze 3 dimensional images/videos, also we used maxpooling and at the end we use flatten layer and a single dense layer with sigmoid as the activation function.

Reason for maxpooling

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn, and the amount of computation performed in the network. Pooling provides the ability to learn invariant features and acts as a regularize to further reduce the problem of overfitting.

**Summary**

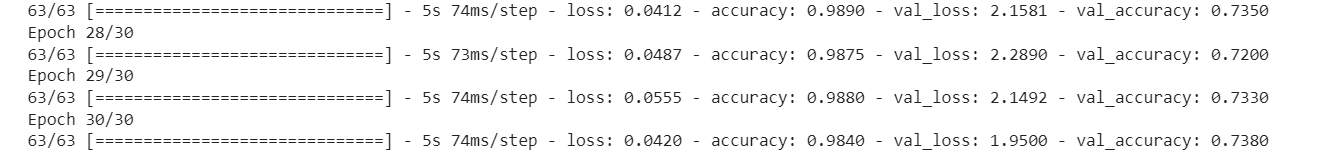
In this experiment, I initially trained the data with basic convolutions and started to see overfitting problems.

Now the question is how to avoid overfitting in Deep Learning (Convolution Neural Networks).

* Adding more training data
* **Dropouts, Regularization, Optimizers (topics learned from first assignment)**
* Cross validation
* Data augmentation
* **Addition of noise to the input data**

**The different approaches tried in this experiment**

With the Basic Convnet:



Chart, histogram, scatter chart

Description automatically generated

**Approach 1:** Added more training data (increased the training samples)

This approach does increase validation and test accuracy but not to a great extent. Here we reduced the overfitting problem.

Table

Description automatically generated

**Approach 2:** usage of dropout layers and Data Augmentation

Using this approach, noticed a big variation in the results, since dropouts play a significant role. The validation and test accuracy has touched 90 percent.

Table

Description automatically generated with medium confidence

**Approach 3:** Optimizer changed to ADAM from rmsprop.

It also increases the results with some variation. Choosing the optimal point (gradient descent) is ADAM optimizer is efficient when compared with rmsprop.

The same approaches applied to VGG16 network.

With the basic VGG16 with initial training samples it does a decent job, probably it is an advantage of pretrained network.

**Using VGG16 Pretrained network**:

**Approach 1:**  Added more training data (increased the training samples)

Yes, it did a great job and I notice a performance improvement in the results.

Table

Description automatically generated with low confidence

**Approach 2:** usage of dropout layers and Data Augmentation

With this approach the performance of the model touches the accuracy of 97 percent with both validation and testing.

Graphical user interface, text, application

Description automatically generated

**Approach 3:** Optimizer changed to ADAM from rmsprop.

Small improvement and the result goes to 98.20.

Table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated