Neural networks notes

1. **Notes on cross\_val\_score**
   1. Use this as a performance evaluation tool of the model we make and mitigate over-fitting.
   2. The main idea is to-
      1. We split our initial dataset, in most case the X\_test from the train\_test\_split.
      2. For each epoch, we partition the data into k folds.
      3. Hold out a set at a time and train the model the rest of the k-1 folds.
      4. Test model on hold out set and retain the evaluation score.

A screenshot of a cell phone

Description automatically generated

* 1. **Types of Cross Validation**
     1. **K-Fold Cross Validation**
     2. **Stratified K-fold Cross Validation**- Splitting of data into folds may be governed by criteria such as ensuring that each fold has the same proportion of observations with a given categorical value. Example- Male or female
     3. **Leave One Out Cross Validation-** Train on N-1 points and test on the Nth point.
  2. **Deciding on a value for k**- k is chosen such that each train/test group of data samples large enough to be statistically representative of the broader dataset. Usually a value of k=10 is common. It is preferable to split the data evenly.

1. To use cross\_val\_score which sklearn and we want to apply a keras model to cross\_val\_score, use kerasClassifier
2. **Dropout Regularization-** To counter overfitting. The idea behind Dropout Regularization is to disable a fraction of the neurons/set their weight to 0 at any given layer. In terms of coding it, we add a dropout layer in between the layers and define the percentage of neurons to disable.
3. **Parameter Tuning-** The idea is to find better values for hyperparameters like number of epochs, batch size, training optimizer (like ‘adam’, etc), activation functions, dropout regularizion, learn rate, momentum. We can achieve this using GridSearchCV which tries all possible combinations of the hyperparameter we want to optimize for. GridSearch does a k\_fold validation on each of the combination of parameter to optimize for.
4. **Great reading for CNN working-** <https://missinglink.ai/guides/keras/keras-conv2d-working-cnn-2d-convolutions-keras/>
5. **Fitting test data for images in CNN- technique- Image augmentation**
   1. The main idea behind image augmentation is to divide the test data in batches and apply random transformations like rotate, flip, etc in order to develop more correlations and hence prevent overfitting by enriching the test dataset in a way
   2. Overfitting usually occurs due to lack of test data and the neural net makes limited number of correlations based on the test data.
   3. Check the code in the keras documentation -> preprocessing -> image processing
6. **Notes on choosing loss functions**
7. **Notes on optimizers**
8. **Notes on choosing activation functions**