**Steps to train MLP:**

1. **Preprocess:** Data Normalization, this make training process faster
2. **Weight Initialization:**

* For Sigmoid/tanh: Xavier/Glurnt
* For Relu: heu
* This should be given last priority, can use weights with Gaussian distribution with small variance.

1. **Choose Activation Function:** Relu is the best choice, we can also use sigmoid or tanh for small networks
2. **Batch Normalization:** must for deep MLP

**Dropout:** It’s also should be done to regularize to avoid overfitting, but you have dropout rate as hyperparameter

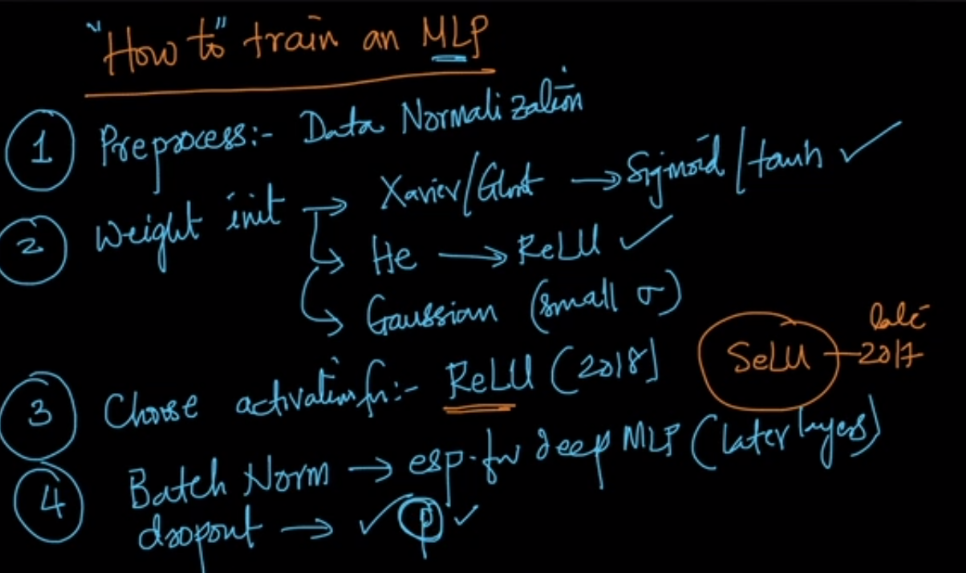
1. **Optimizer:** Best Choice is Adam, along with this Adadelta can be also be used.
2. **Hyperparameters:**

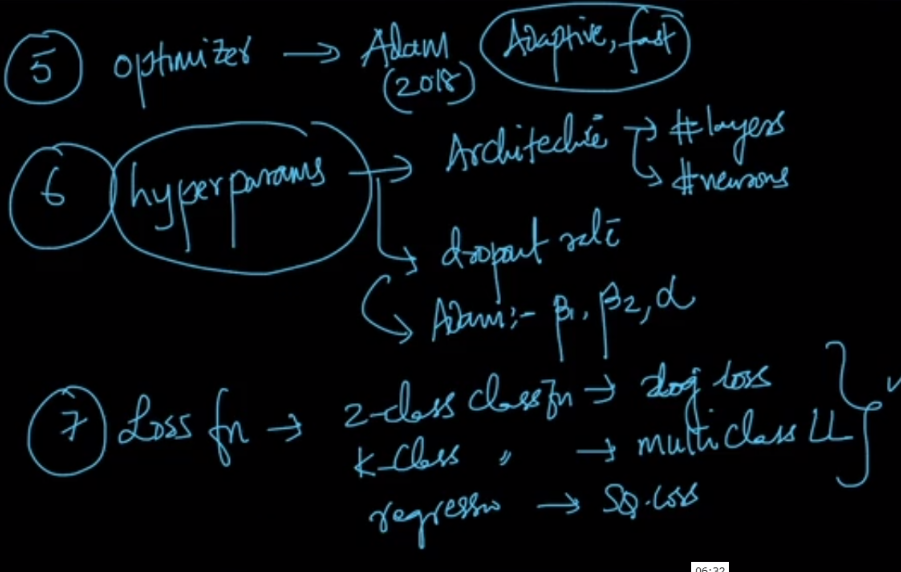
* **Architecture:** No. of layers, No of neurons or unit in each layers (One rule of thumb is to have fewer neurons in later layers as compared to earlier layers).
* **Dropout rate**
* **Adam:** b1, b2, alpha

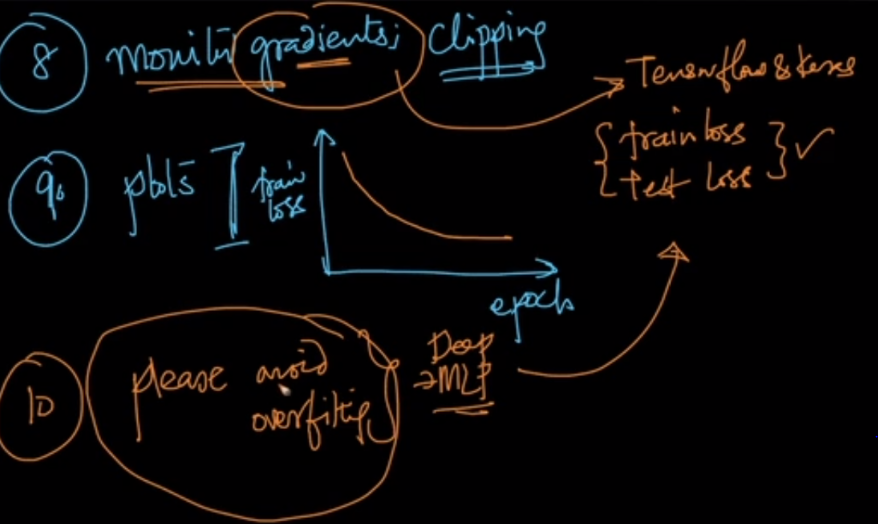
1. **Loss function:**

* **For 2 class classification:** Log loss
* **For k-class classification:** Multic class log loss
* **For regression:** square loss

1. **Monitor gradient and perform clipping if gradient exploding happening**
2. **Plot loss and epoch graph which tells whether model is training right or not**
3. **Avoid Overfitting:** Plot train and test loss if they are coming closer and decreasing that means model is not overfitting.







To know more about hyperparameter tuning: <https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-i-hyper-parameter-8129009f131b>