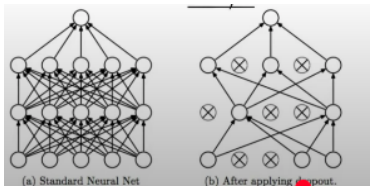
- 784-128-64-10
- during each forward - randomly turn off neurons - simplifies the NN
- Since har forw pass m new NN use kr rhe in a way - regularisationish

- Applied to hidden layers ; APplied after the ReLU activation function (set p=0.5 50% neurons turned off - hyperparamet h)
- This has a regularization effect
- During evaluation dropout isnt used.
- Just one argument p

```
class MyNN(nn.Module):
    def __init__(self, num_features):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(num_features, 128),
            nn.ReLU(),
            nn.Dropout(p=0.3)
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(p=0.3)
            nn.Linear(64, 10)
        )

    def forward(self, x):
        return self.model(x)
```



BATCH NORMALISATION (nn.BatchNorm1d) https://www.youtube.com/watch?v=2AscwXePInA&t=1543s

- Improves Training stablity - NN have a problem during training (**Internal Covariant Shift)**

💡 **Internal Covariant Shift:**

each layers o/p is i/p for next.

- The value of wts are constantly being changed - due to this the distribution of the input which a layers gets from the prev layer also changes - training unstable
- So during every mini batch apply batch norm - activations ko norm kro (in a given set of range)

- Applied to hidden layers
- Applied after Linear layers and BEFORE activation functions

💡 How are the Activations Normalised?

- Computes the mean and variance of the activations withina mini batch and uses these stats to normalize
- Includes some learnable parameters - gama(scaling) adnd beta(shifting) which allows the network to adjust the normalised outputs

- Recues the ICS , stabs the training process and **allows use of HIGHER LR**
- Regularisation effect - because noise is introduced in system - har mini batch ka alg mean and var
- nn.BatchNorm1d(no.of neurons)

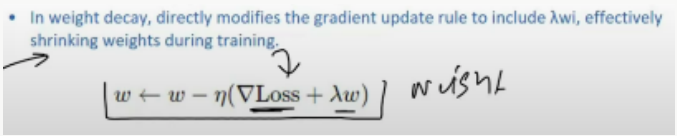
L2 Regularization https://www.youtube.com/watch?v=4xRonrhtkzc&t=1452s

$$\text{Loss}_{\text{reg}} = \text{Loss}_{\text{original}} + \lambda \sum w_i^2$$

- Basically penalises the larger weight vales and ecourage smaller , more generlisable wts
- Extra term is added in the loss function (penalty) ; in L1 there is mod(Wi) instead of sqaure
- Now goal is to minimize the combination of these - weights dont get v high values - overfitting reduces
- Applied to Model Weights not biases

💡 Weight Decay(easist way to apply L2)

In pytorch we can directly apply this to the optimisation step - druing grad descent direct gradiant mai loss ko add krdo



- Large weight values are distributed to other weights
- Hyperparametr - lamda(reg coeff) - pass in the optim parameter itself .

- No effect on testing

https://colab.research.google.com/drive/1YDVmsVD8zkdDh5lqumA_Htlh_WqH10FC?usp=sharing#scrollTo=0UpCVk9X-JaI

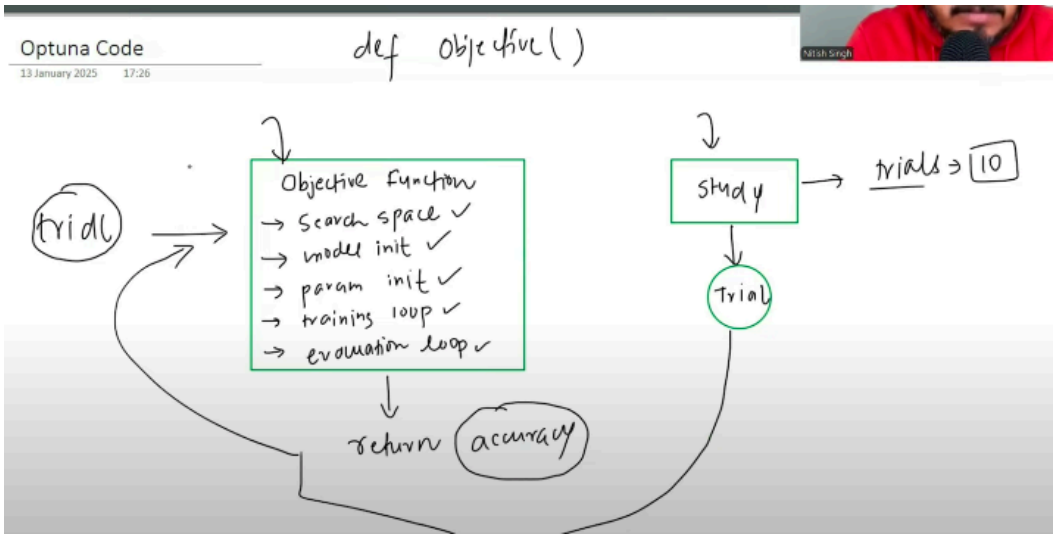
Hyperparameter Tuning ANN using Optuna

- How do we decide how many layers how many neurons - no logic; but how do we know kitne lene h
- Hyperparameter - epochs,Optimiser, batch_size , lr , drop percent, lamda(wt decay)-reg , number of hidden layers , no. of neurons — Experimentation

- GridSearch CV , RandomSearch CV, ***Bayesian Search(Most advanced way) - Using Optuna***

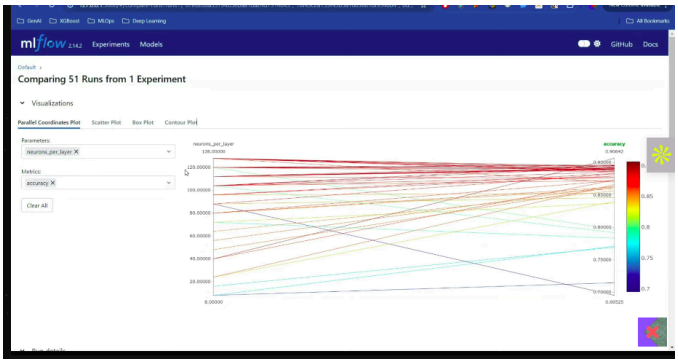
- If u wanna learn basics of optuna - https://www.youtube.com/watch?v=E2b3SKMw934
-

> Overview of optuna: - u hv to make a objective function with trial object as input, def objective(trial)
>
> - Define Search Space(lowest and highest space , created for every hyperparameter) ; model init (class and object) ; param init (epochs , lr , ,lsso , optim) ; trinaing loop ; evaluation loop
>
> - returns accuracy)
> - Study Object - no, of trials is defining



https://colab.research.google.com/drive/11n3IF779Ix5b0E_cafFbSR_D2aRbZ1oN

- After trials - try increase number of trials , increase search space .
- Optuna x MLFlow - tracks the trial runs - you can see which parameter value are more tried out(better) - in industry



Building a CNN (pytorch)

- When working on image based dataset ANNs arent that good ; CNNs were created to train on images and videos.