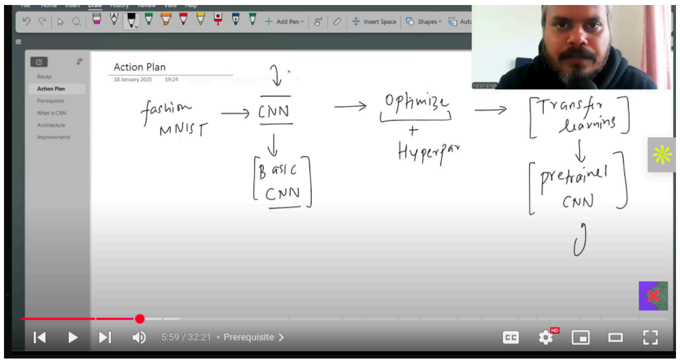
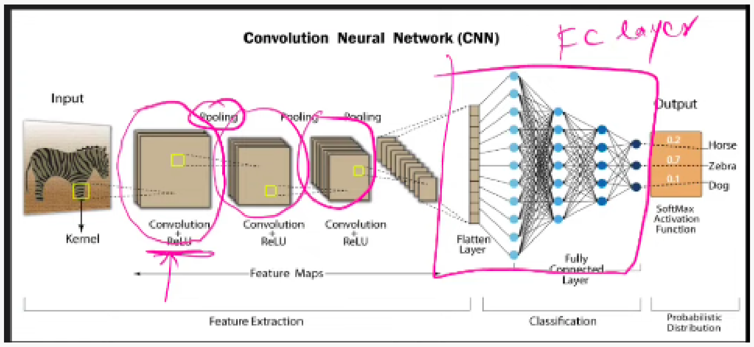


CNN architecture (100 days deep learning) - https://youtu.be/hDVFXf74P-U?si=aGyBnSTfv_RPQzDw

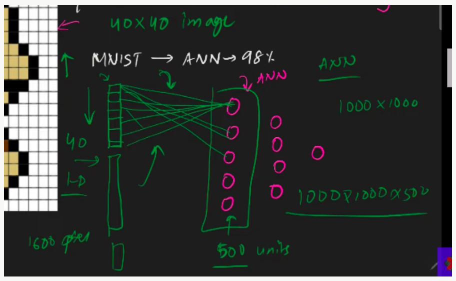
Theory for CNN (notes)

CNN , covnets are a special kinf od NN for processing data that has a known gird-like topology like Time series data (1D) or images (2D).

- Convulation layes perform covulation (ANN mai matrix mult use krte h idhar convulation)
- Convulation layer - Pooling layer - Fully connected layer



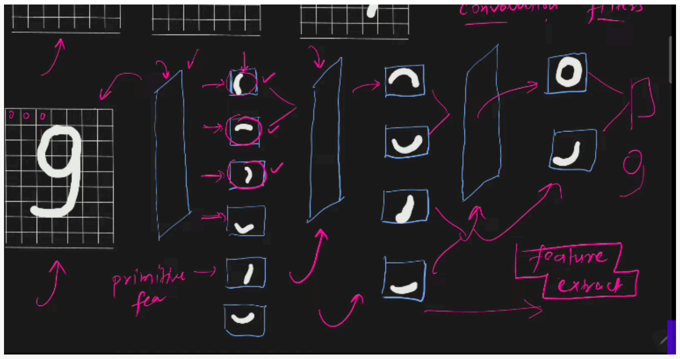
- Why not just use ANN? - High computation cost(layers mai 2d to 1d krke daaldna pdega, weights bhot zda !!!, pehli layer mai hi 1lakh wts) , overfitting(har minute cheeze capture krne ka try) , loss of imp info like spatial arragemnet of pixels(like distance between nose and eyes on monkey , which is lost when 2d → 1d) ; (always cnn > ann for image)



CNN Intuition

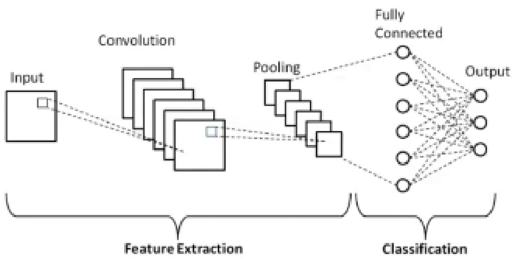
Basically breaks down the image into feature (eg. image of nine has a circle then 2 lines , edges etc) - makes more complex features layer by layer

- COnvulation layers - feature extractor - mathematic operations bascially- move the filter over the image- feautres from prev layer are merged to create more meaningfull feature for the next layer.

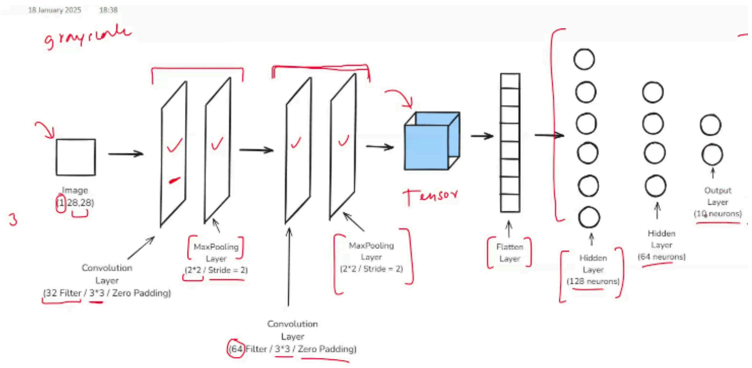


- Tasks - image classification ; Object localisation ; Object detection and localisation ; Face detection and recog ; Resolution upgrader ; BnW to color ; Pose recog

TBC!!!



- Pooling layers main feature is reduce size of tensor,

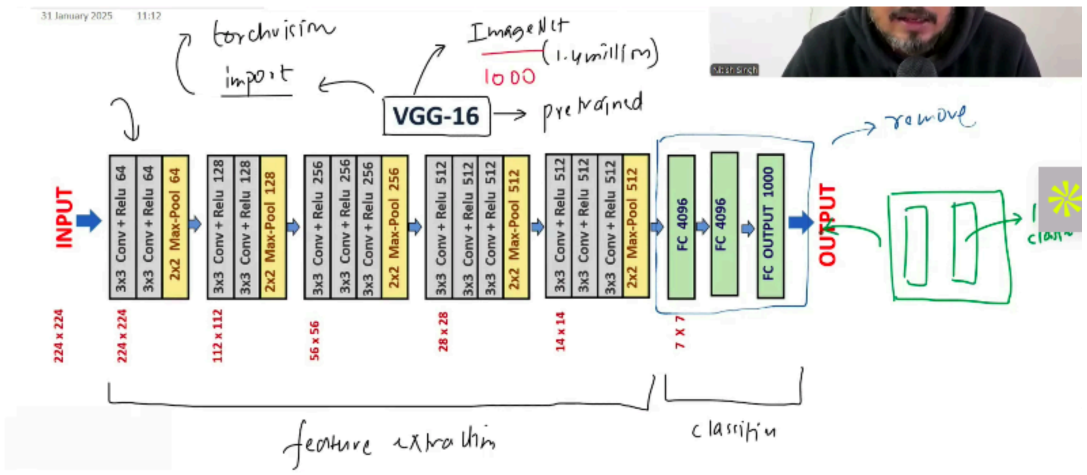

18 January 2025 18:38
grayscale
Image (1,28,28)
3
Convolution Layer (32 Filter / 3*3 / Zero Padding)
MaxPooling Layer (2*2 / Stride = 2)
Convolution Layer (64 Filter / 3*3 / Zero Padding)
MaxPooling Layer (2*2 / Stride = 2)
Tensor
Flatten Layer
Hidden Layer (128 neurons)
Hidden Layer (64 neurons)
Output Layer (10 neurons)

Building CNN - https://colab.research.google.com/drive/1r7kTcDb27ds6eJl4u-FLTNPjTkbc6A_a (overfitting - 92 , 97)

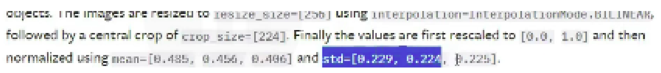
Improved version (data aug , dropout , optuna written) - https://colab.research.google.com/drive/1AWdxO9qNBpzdQlt-xSznFmr5k9e-uUTz?usp=sharing (93.1 acc)

Transfer Learning (VGG16)

- Instead of traning a model from scratch , comp expensice and req large datasets. TL leverages knowledge from a pre-trained model to improve learning effieciency and performace.
- Fine tuning for a new task - training just a classification head and maybe unfreeze some layers of the pretrained model.
- These pretrained model are imported from torchvision.models



- We need to dettach the classification head as 100 output classes nai chaiye
- VGG expected PIL RGB images (batch , chaneel , H , W) AND resized to 256,256 and crop to 224,224 (center ke) THEN pixel value is scaled to 0,1 and normalised with certain provided means and stnd devs.



objects. The images are resized to `resize_size=[256]` using `interpolation=InterpolationMode.BILINEAR`, followed by a central crop of `crop_size=[224]`. Finally the values are first rescaled to `[0.0, 1.0]` and then normalized using `mean=[0.485, 0.456, 0.406]` and `std=[0.229, 0.224, 0.225]`.

- Reshape (28,28) → 2d (matrix)
  datatype→np.uint8 (PIL format)
  1d to 3d
  tensor → PIL Image
  Resize to 3,256,256→ center crop 3,224,224
  tensor(scale) → 0,1
  normalize(all 3 channels ko sep normalise krna h) - basically subtract mean and divide stnd dev

- These transformation are v easy with torchvision.transforms