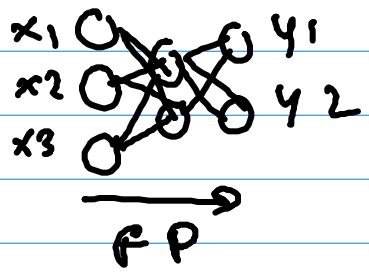
upto 8th week

W1 → Basic Python / External module

W2 → How Neural Networks train
- Forward Propagation $y = \sum(wx) + b$
- Error



W3 → Optimizers
- Activation functions
- FFNN
- In class Project I
- Iris Flower / Digits

W4 → CNN
- Kernels, convolution
- Feature extraction
- Cat Dog classification

W5 → Nvidia Self driving car

W6 → TensorFlow object Detection API

W7 → RNN, LSTMs
- Stock market value Prediction

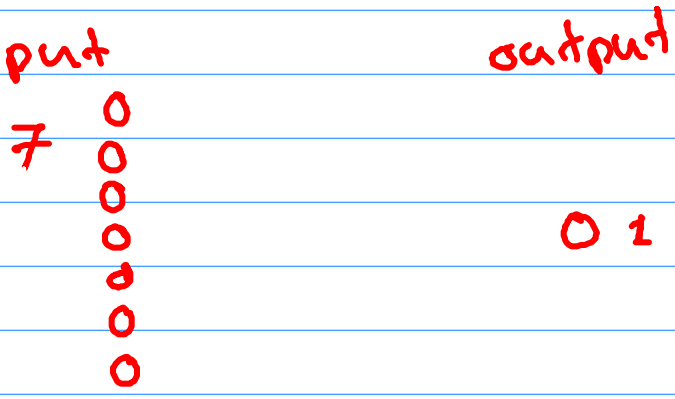
DL

- Supervised DL
  - Forward Propagation
  - classification / Regression
  - Gradient Decent
  - Back propagation
  - Activation, Error, optimizers, Dropout, overfitting
  - FFNN, CNN, RNN
- unsupervised DL
  - Autoencoders
  - Generative Adversarial Networks
- Reinforcement DL
  - Deep RNN

dataset ⟶ 8 cols (0-7 features, 8 label)

* labeled data ⇒ supervised DL
* FFNN (given)
* Label (0-100) → Regression Problem
  ↳ Loss - MSE

* NN Architecture



Mean Squared Error

$$E = \frac{1}{m} \sum (y_A - y_P)^2$$

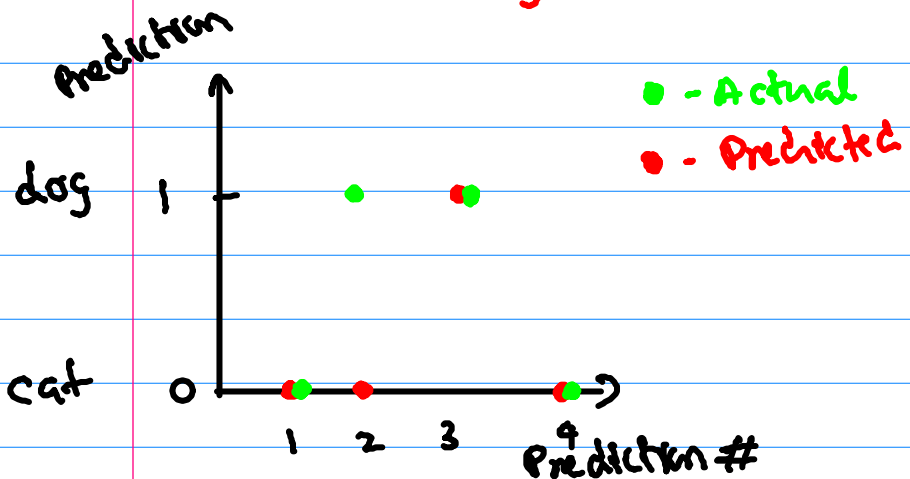
target should be normalized !

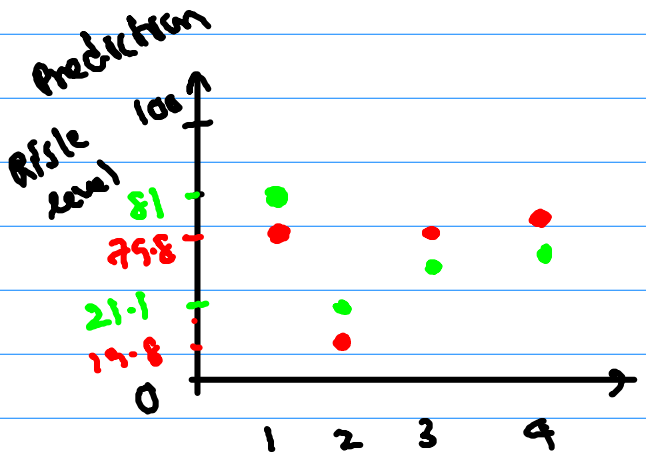
without normalizing $E = (30 - 20)^2 = 100$

normalized $E = (0.3 - 0.2)^2 = 0.01$

classification → Accuracy

Regression → R2 score





$$Acc = \frac{\#\text{correct predictions}}{\#\text{total Predictions}}$$

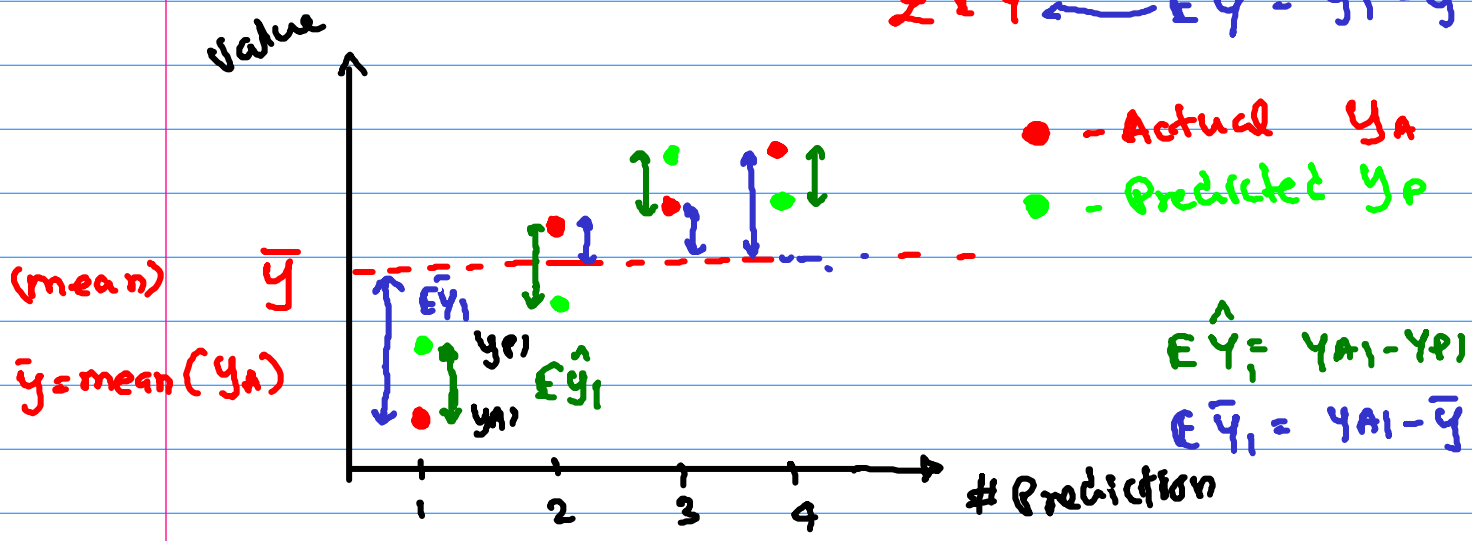
$$Acc = 3/4 = 0.75$$

R2 Score

$$R2 = 1 - \frac{\sum E\hat{y}^2}{\sum E\bar{y}^2}$$

$E\hat{y} = y_i - \hat{y}$

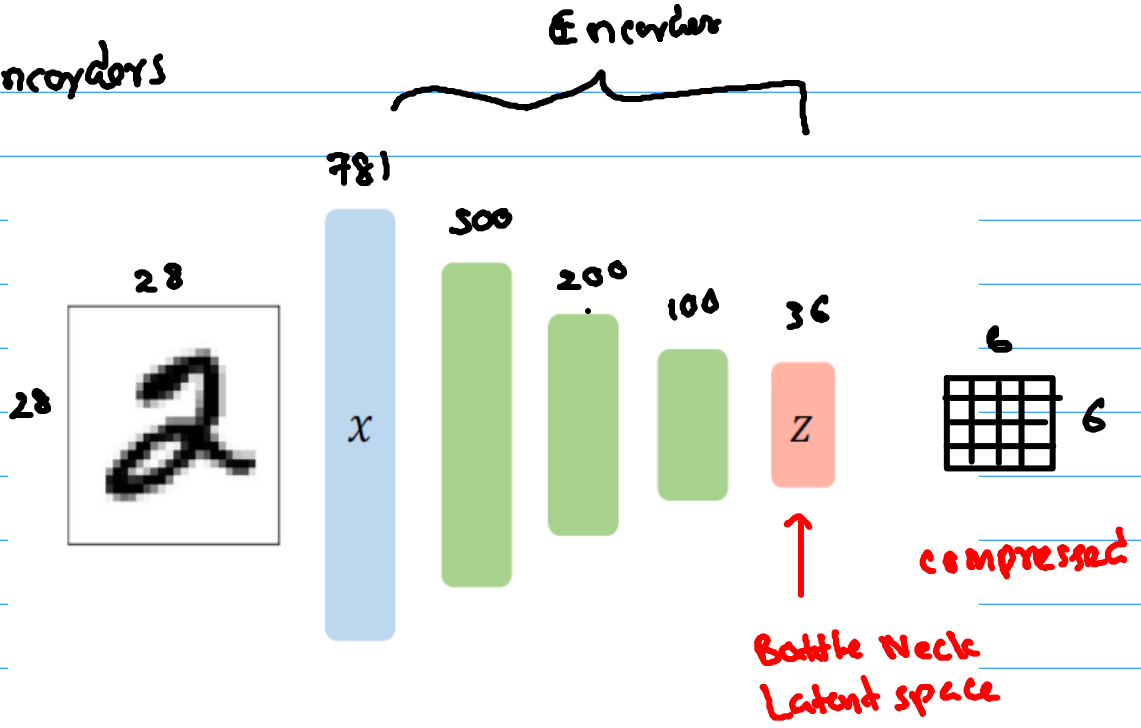
$E\bar{y} = y_i - \bar{y}$



$$(E\hat{y})^2 = E\hat{y}_1^2 + E\hat{y}_2^2 + E\hat{y}_3^2 + E\hat{y}_4^2$$

$$E\bar{y}^2 = E\bar{y}_1^2 + E\bar{y}_2^2 + E\bar{y}_3^2 + E\bar{y}_4^2$$

Auto Encoders



Supervised DL

$$Error = F(y_{actual}, y_{predicted})$$

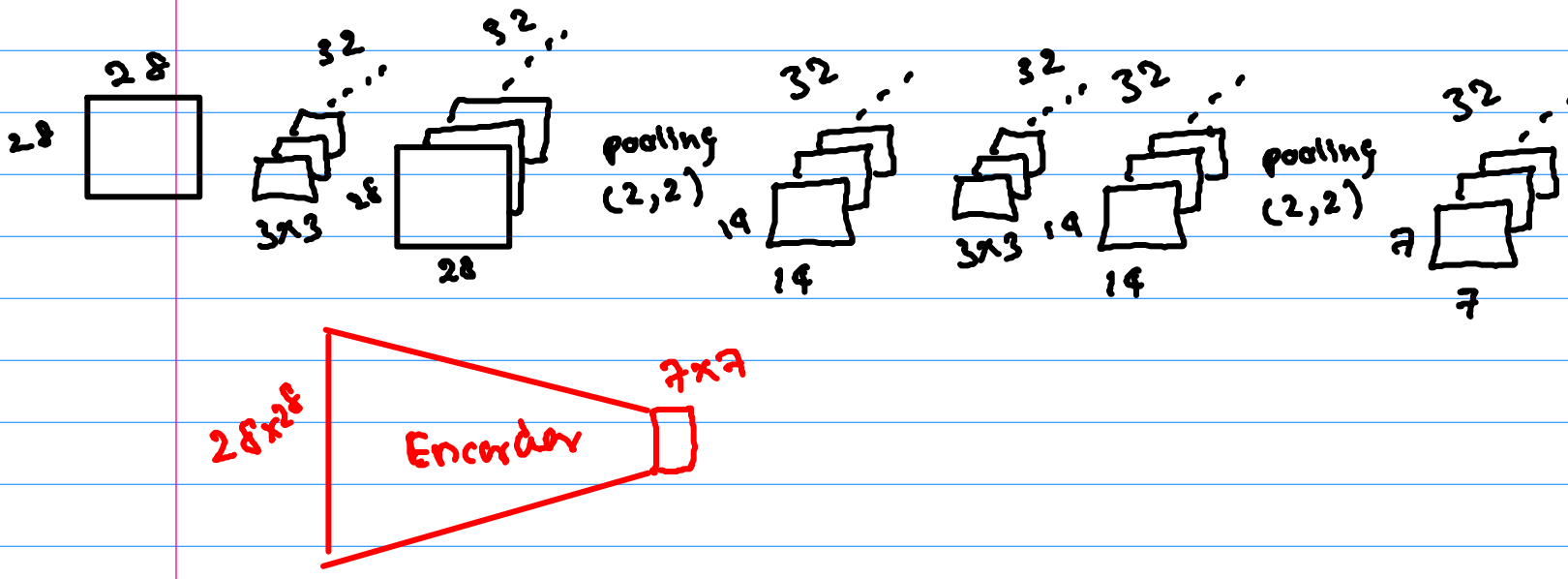
x - Features
y - labels

Unsupervised DL

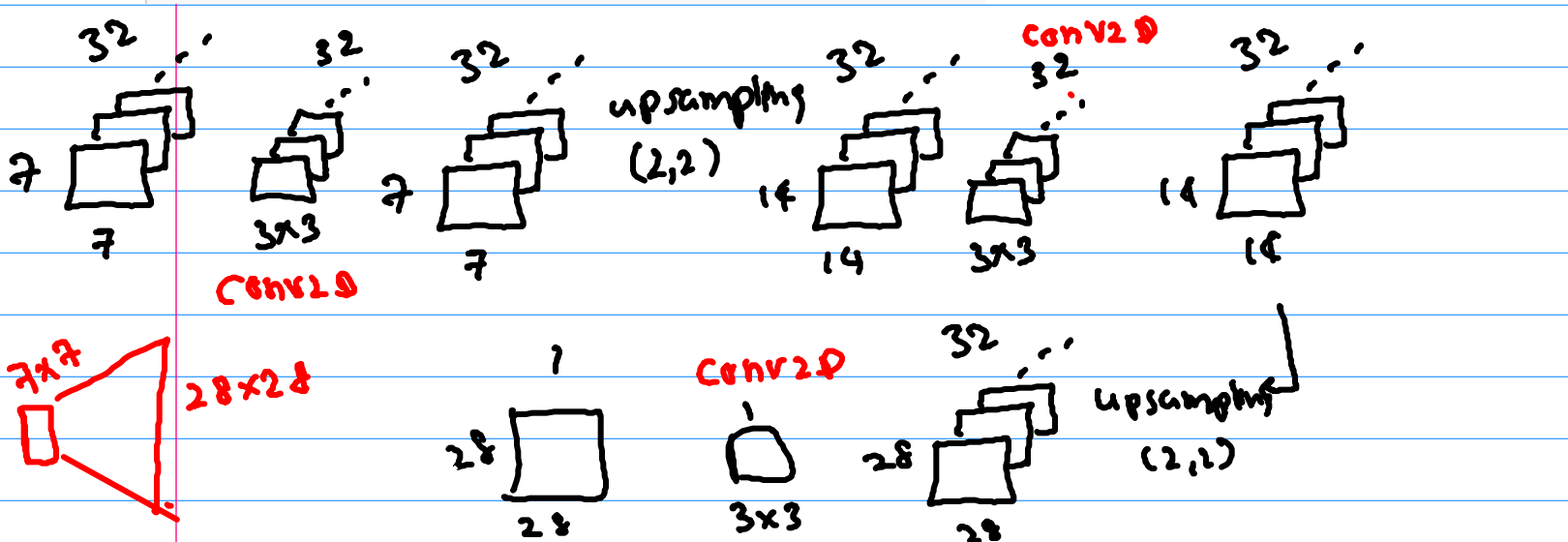
$$Error = F(x, \hat{x})$$

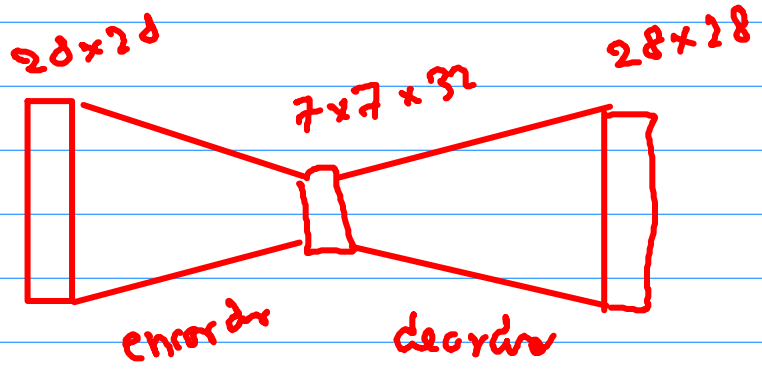
Autoencoder for Noisy MNIST

```
from keras.layer import Input,Dense,Conv2d,MaxPooling2D,UpSampling2D
from keras.models import Model
from keras import backend as K

input_layer=Input(shape=train_data.shape[1:])

model=Conv2D(32,(3,3),activation='relu',padding='same')(input_layer)
model=MaxPooling2D((2,2),padding='same')(model)
model=Conv2D(32,(3,3),activation='relu',padding='same')

encoder=MaxPooling2D((2,2),padding='same')(model)
```




```
model=Conv2D(32,(3,3),activation='relu',padding='same')(encoder)
model=UpSampling((2,2),padding='same')(model) #inverse of pooling
model=Conv2D(32,(3,3),activation='relu',padding='same')(model)
model=UpSampling((2,2))(model)

decoder=Conv2D(1,(3,3),activation='relu',padding='same')(model)
```




28x28
7x7x32
28x28
encoder
decoder