Block A

Training Stability

↓

Initialization / Optimizers

Block B.

Vision models

↓

CNN's Conv, Pooling, architecture.

Neural Network Parameter Initialization.

Training fails → vanishing gradients
→ Exploding gradients
→ Symmetry problems.

Initialization decides how signal flow forward & Gradient flow backwards.

(1) Zero Initialization

all weights = 0.

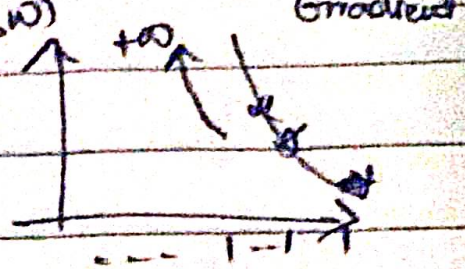
all neurons → Same output
all Gradients are Identical
neurons never Specific ✗

} Bias = 0 is fine
weights = 0 is deadly

(2) Random Initialization → Small random values.
(normal or uniform).

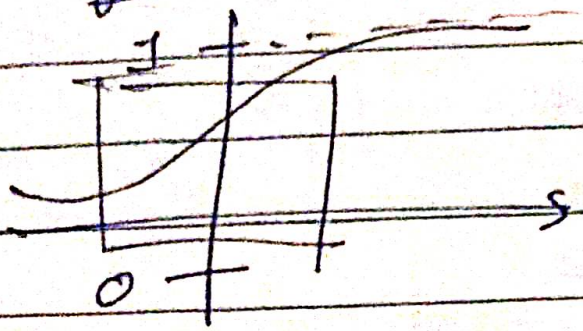
Problems: Too Small → gradients vanish
too large → gradients explode

Exploding Gradient

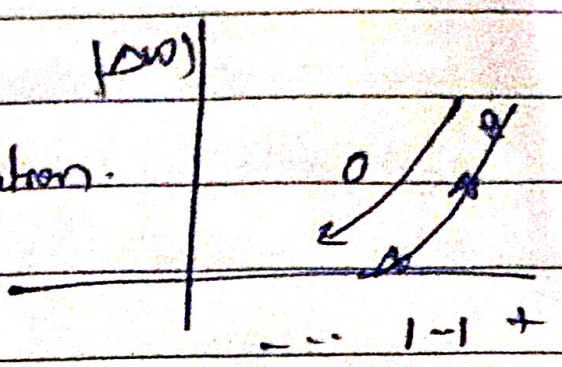


(3) Xavier Initialization (Glorot):

Sigmoid



→ Linear Combination.



designed For Sigmoid / Tanh

vanishing Gradient

$$Var(w) = \frac{1}{n_{in}}$$

→ Output Signal Should'n Shrink or blow up as depth Increases.

✗ Bad For ReLU.

✓ Great for tanh / Sigmoid.

(4) Kaiming Initialization (He).

$$\sigma = \sqrt{2.0/n_{in}}$$

Designed for

(ReLU) / Leaky ReLU

$$Var(w) = \frac{2}{n_{in}}$$

ReLU → Kills half the activation (zeros).

Rule of thumb

ReLU → Kaiming

Tanh → Xavier.

Sigmoid → xavier

Tanh → xavier

ReLU → Kaiming

Deep CNN's → Kaiming.

Optimizers → ADAM V/S ADAMW.

→ Gradient decent problem.

→ Same learning rates for all parameters

→ Sensitive to Curvature

→ Slow Convergence.

ADAM = momentum + RMSprop

① First moment → mean of gradients

② Second moment → variance of gradients.

Why worked

Fast Convergence

Adaptive learning rate per parameter

Robust to noise gradients

Hidden problem.

Adam + L2 regularization ≠ true weight decay.

AdamW → (modern default)

weight decay is applied directly, not via loss.

→ proper regularization

→ Better Generalization

→ used in Transformers, CNN's, VIT's.

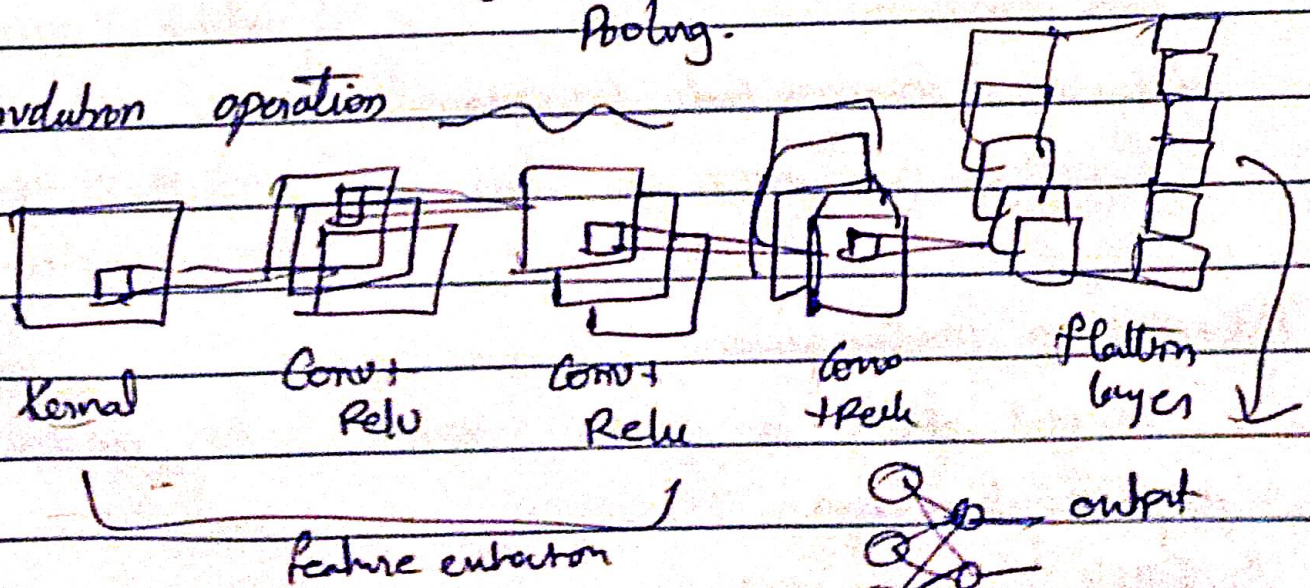
ADAM Adapts Gradients
ADAMW fixes Regularization

optimizer Summary:
SGD → Theory, Small models.
Adam → fast Convergence
Adamw → production + research default

Block B → CNN's Convolutional Neural Networks.
But Connected nets:
→ Ignore Spatial Structure
→ Too many Parameters.

CNN's Exploit:
① Locality
② Translation Invariance.
③ weight Sharing

Convolution operation



weight sharing = fewer parameters + translation equivariance + less overfitting

weight sharing → same filter everywhere → learns edges, textures, patterns.

Small filter slides over Image

Dot product → feature map

Parameters → kernal size eg (3×3)
→ Stride
→ Padding.

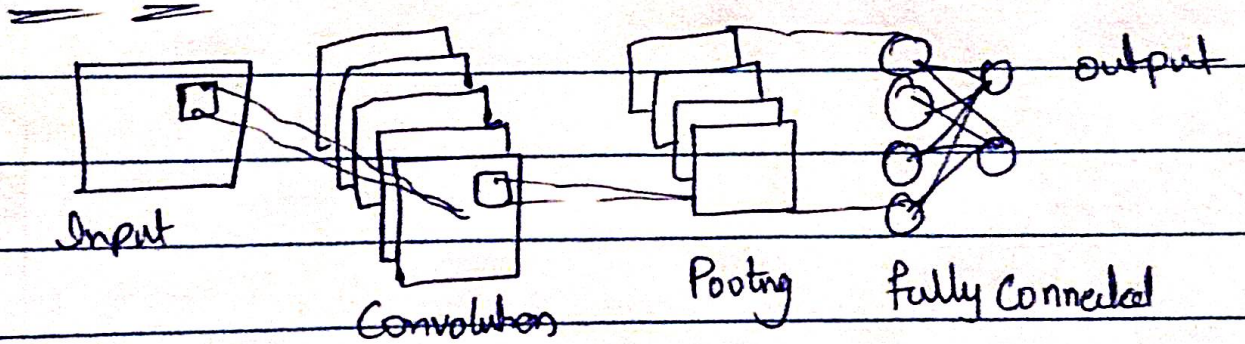
Pooling → → pooling reduces spatial size

maxpooling, Average pooling

→ makes model robust to small shifts

effects → ① Downsampling
② Slight Information loss (Intentional).

CNN Flow



feature extraction

classification.

Input
↓
conv → ReLU
↓
Pool
↓
Conv → Relu.
↓
Pool

↓
Flatten
↓
Fully Connected.
↓
Softmax

early layer → edges.
mid layer → Shapes
Deep layer → objects

CNN's → Feature extractor + classifier

Conv + pool = feature extractor

FC layers = decision maker.