Convolution from First Principles

28 December 2025 04:56 PM

why MLP failed?

- Locality Destruction
- No Parameter sharing
- Parameter Explosion

How we Solve MLP's Problem?

⇒ we apply Mathematical Constraints

Mathematically

a) Locality Destruction

each MLP layer performs,

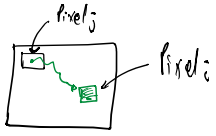
$x \in R^{H \times W \times C}$; $W \in R^{d \times (H \times W \times C)}$

$y = Wx + b$ → This is global linear operator over the image

$$\Rightarrow y_i = \sum_{j=0}^{N} x_j + b_j$$

meaning

⇒ right now pixel$_i$ is influencing pixel$_j$ and they are not even close (neighbour).



Notes: 1) Right now each x_j is contributing for the prediction of y_i → images need local pixels to only talk past strongly

2) MLP is permutation-sensitive but Geometrically Blind.

b) No Parameter sharing

$$y_i = \sum_{j=0}^{N} W_j x_j + b_j$$

assume:- two pixel x_p and x_q represents same local pattern at two different location

Because Right now there is no parameter sharing.

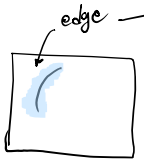
In General, $[W_{x_p} \neq W_{x_q}]$ ← model will have to re-learn this pattern at different location.

← Computation is wasted.

Note: In current Configuration,

1) Model must relearn the same feature at every different position

2) Translation of a pattern produce different representation.

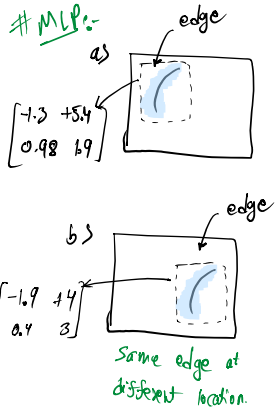


No matter where this curvy edge is we want same representation for this curvy edge.

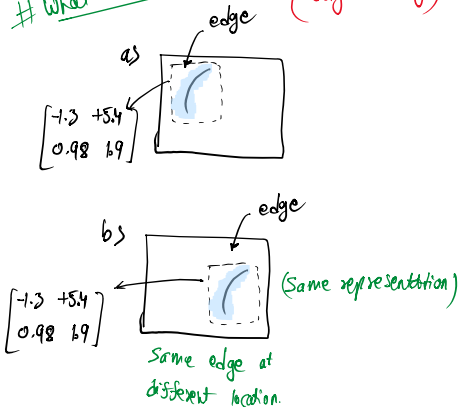
problem

The same local pattern at two different location is processed by different linear maps.

MLP:-



What we want (Convolution) (weight sharing)



c) Parameter Explosion

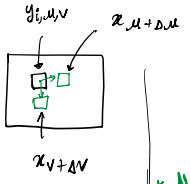
MLP → Parameter Count for Single Layer in MLP,

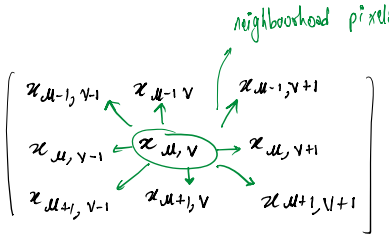
$|\theta| = (w \times h \times C \times d)$

Minimal Constrain on Linear Operator

1) Locality Dependent

$y_{i,u,v}$ must be dependent only on neighbour pixels → $x_{u+\Delta u, v+\Delta v}$





Notation →

$A_{i,u,v,c,u',v'}$ ≡ weight connecting input signal (c,u',v') to output signal (i,u,v).

⇒ $A_{i,u,v,c,u',v'} = 0$ unless $|u'-u| \le K$ and $|v'-v| \le K$.

→ Pixel that are not in a range of "KxK" will not contribute for predicting output signal.

⇒ Define offset, $\Delta u = u'-u$, $\Delta v = v'-v$

So operator Becomes,

eqn ① →

$$y_{i,u,v} = \sum_{c=1}^{C} \sum_{\Delta u, \Delta v} A_{i,u,v,c,\Delta u,\Delta v} \cdot x_{c,u+\Delta u, v+\Delta v}$$

→ locality Achieved But weights are still dependent upon absolute position (u,v).

2) Weight sharing

↳ we want to store the same weight across a range of "KxK" → 'offset'.

#4.

Define Constrain.

equivariance

X → input space, Y → output space.

G: group acting on both input and output spaces

$T: X \to Y$

if → * $T(gx) = gT(x)$; $\forall x, g$

then 'T' is equivariant to 'G'

for Images. $G \to g \to \tau_s \to$ Spatial shift.

⇒ $T(\tau_s x) = \tau_s T(x)$ → this will force weight sharing

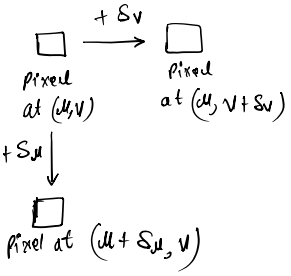
How Do we shift the image!?

→ we define "τ_s" → shift the image.

⇒ $S = \{S_u, S_v\}$

S_u → up/Down shift, S_v → Right-left shift

We want,



$$(T_s x)_{c,u,v} = x_{c,u-S_u,v-S_v}$$

→ pixel at (u,v) is just a shifted version of pixel at [illegible]

pixel at (u,v) is just a shifted version of pixel at $((u-s_u),(v-s_v))$

what we have now,

$$y_{i,u,v} = \sum_{c=1}^{C} \sum_{\Delta u,\Delta v} A_{i,u,v,c,\Delta u,\Delta v} \cdot x_{c,u+\Delta u,v+\Delta v}$$

shift input by s

$$\Rightarrow T(\tau_s x)_{i,u,v} = \sum_{c=1}^{C} \sum_{\Delta u,\Delta v} A_{i,u,v,c,\Delta u,\Delta v} \cdot x_{c,u+\Delta u-s_u,v+\Delta v-s_v} \rightarrow \text{LHS.}$$

$$\Rightarrow \tau_s(T(x))_{i,u,v} = \sum_{c=1}^{C} \sum_{\Delta u,\Delta v} A_{i,u-s_u,v-s_v,c,\Delta u,\Delta v} \cdot x_{c,u+\Delta u-s_u,v+\Delta v-s_v} \rightarrow \text{RHS}$$

But, $T(\tau_s x) = \tau_s(T(x))$

$$\sum_{c=1}^{C} \sum_{\Delta u,\Delta v} A_{i,u,v,c,\Delta u,\Delta v} \cdot x_{c,u+\Delta u-s_u,v+\Delta v-s_v} = \sum_{c=1}^{C} \sum_{\Delta u,\Delta v} A_{i,u-s_u,v-s_v,c,\Delta u,\Delta v} \cdot x_{c,u+\Delta u-s_u,v+\Delta v-s_v}$$

$$\Rightarrow A_{i,u,v,c,\Delta u,\Delta v} = A_{i,u-s_u,v-s_v,c,\Delta u,\Delta v}$$

(original position; shifted position)

→ This solution enables weight sharing.

Because weight at u,v is used at $(u-s_u, v-s_v)$.

$\Rightarrow$ Independent of u,v.

$\Rightarrow$ weight don't depends on absolute position

Meaning

$$A_{i,u,v,c,\Delta u,\Delta v} = A_{i,u-s_u,v-s_v,c,\Delta u,\Delta v}$$

$\Rightarrow$ The weight used at output position (u,v) is exactly the same as the weight used at output position $(u-s_u, v-s_v)$ as long as the relative offset $(\Delta u, \Delta v)$ is the same.

so if,

$$A_{i,u,v,c,\Delta u,\Delta v} = A_{i,u-s_u,v-s_v,c,\Delta u,\Delta v} \quad \text{eq}^{n0}\,(2)$$

$\Rightarrow$ it wasn't depends on (u,v); Because it satisfies eq^{n0} (2)

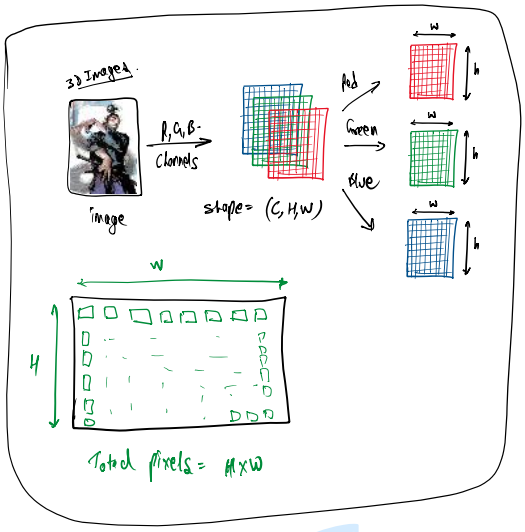
$\Rightarrow$ rename it as, (Convolution Kernel)

$$A_{i,u,v,c,\Delta u,\Delta v} = K_{i,c,\Delta u,\Delta v}$$

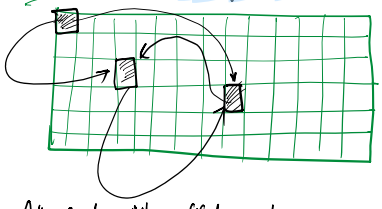
Before V/s After

Before (MLP)

- Image



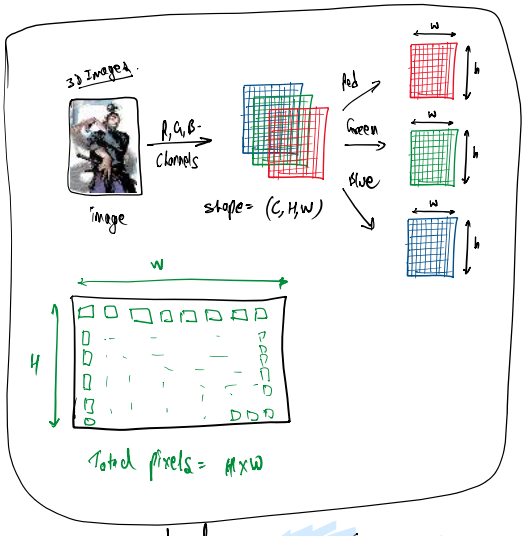
- Co-ordinate dependent linear Transformation



$$y_i = \sum_{i=0}^{N} w_i x_i + b_i$$

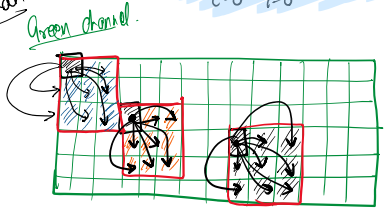
→ All pixel will effect each
→ each one have different weight.

After (Convolution)



- Translation-equivariant local linear Transformation

$$T(g(x)) = g(T(x))$$



$$y_i = \sum_{c=0}^{C} \sum_{i=0}^{N} K_{i,c,\Delta u,\Delta v} \cdot x_{c,\Delta u,\Delta v}$$

→ Pixel only effects the pixels within the red '3x3' window size (Kernel size)
→ only local pixels influence the output
→ distant pixel have zero contribution

$A_{i,u,v,c,u',v'} = 0$; unless $|u'-u| \le K$ and $|v'-v| \le K$

→ weights are shared across spatial location not across offsets.

Conclusion

Convolution

$$y_{i,u,v} = \sum_{c=1}^{C} \sum_{\Delta u=-K'}^{K'} \sum_{\Delta u=-K'}^{K'} K_{i,c,\Delta u,\Delta v} \cdot x_{c,u+\Delta u,v+\Delta v} + b_i$$

or

$$y_{i,u,v} = \sum_{c,\Delta u,\Delta v} K_{i,c,\Delta u,\Delta v}\, x_{c,u+\Delta u,v+\Delta v} + b_i$$

MLP

$$y_i = \sum_{j=1}^{CHW} W_{i,j}\, x_j + b_i$$

Constrained → locality, weight sharing

⇒ A Convolution is just a linear Layer

⇒ A Convolution is just a linear layer
whose weight Matrix has been constrained
to be sparse (local) and structured (shared across spatial shift)