St. Paul's Mission School
Kolkata

NIT. ROURKELA
WINTER.
INTERNSHIP.
DIARIES.

Name Kushanava Rakshit.

Class 2nd Sec. B(4) Roll No. 2022CEB008.

Subject NIT. ROURKELA Year 2024-25

Expt. No. ---------- Date ----------

Page No. ----------

EMG signals

electromyography is a technique used to measure the electrical activity produced by muscles.

Muscle activity generates electrical signals called action potentials (APs), detected by non-invasive techniques like placing copper electrodes on the skin.

EMG output data is a superimposition of all the APs coming from all the motor units that the electrode is detecting.

CNN detects APs and then patterns in the APs, so features are automatically detected from the output data.

Henneman's law

Motor units are recruited from smallest to largest based on the size of the motor neuron and the associated muscle fibres.

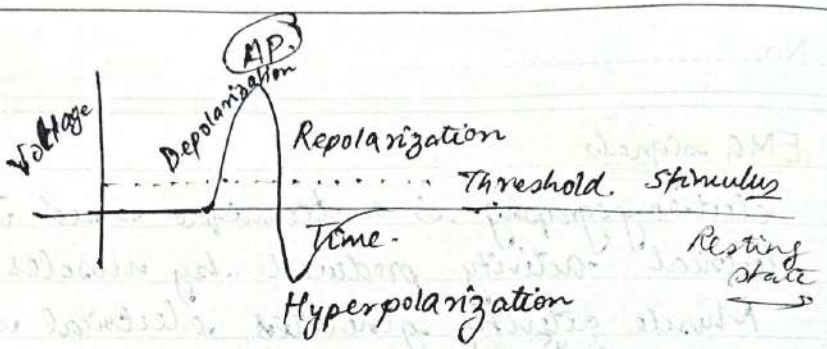
↓

Extracting information about the type of muscle fibres that are firing, would give an insight on the amount of force we need to produce in our arm.

(APs of different muscle fibres have a certain range of frequencies)

Teacher's Signature : _____

EMG output data is noisy hence requires filtering and feature extraction to get useful information.

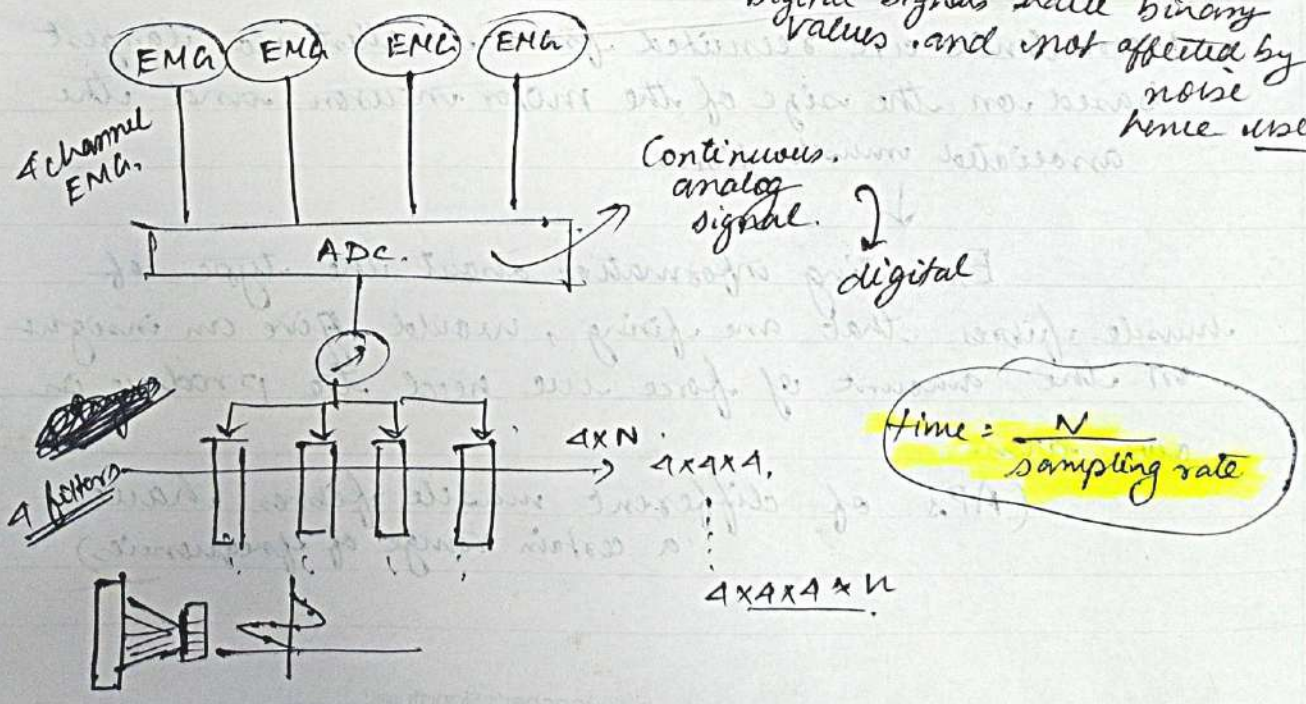


(1) Patterns in APs.
(2) Features of APs.
(3) Location from where signal was taken

→ Determine corresponding motion.

Force required ↑.
AP frequency ↑.
Amplitude ↑.

Digital signals have binary values and not affected by noise hence used

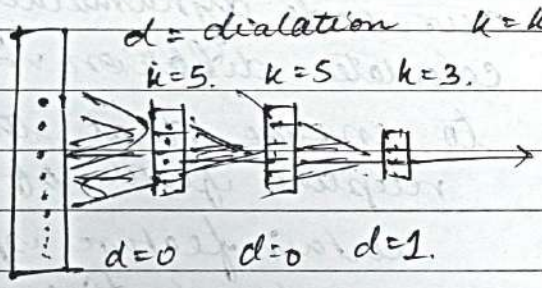


Time = N / sampling rate

4 filters are created.

Each filter with 3 layers
with each layer having
different dilations.
to capture data.
with a larger receptive field

Type I Motor units (slow twitch) → Low frequency
low amplitude AP.



R = sampling rate / frequency $= 1 + \sum_{i=1}^{L} (K_i - 1) \cdot d_i$.

L = no. of layers.
R = Receptive field
K_i = Kernel size of ith layer.
d_i = dialation state of ith layer.

3 features for AP.
Type I. 30Hz
Type II. 100Hz.
Type IIb. 225 Hz

1 feature for silence $k = 5$.

(NO RELU in 1st layer). { To capture negative parts of AP }

Teacher's Signature :

Type I.: Freq 30 Hz
$k = 15$.
$L = 4$.
Stride = 1.

Trainable features
i.e. kernel values are calculated by back propagation.

Sampling rate → set as a variable in code so that the code can be adapted for different hardwares

$$d_0 = \frac{\text{sampling rate}}{\text{freq} \times k}$$

∴ size of AP doesn't change our code dynamically calculates dilation values. to ensure that the receptive field of a certain feature spans a fixed time period.

Type II a. $k = 10$.
Freq = 100 Hz.
$L = 4$,
stride = 1
∴

Trainable feature
dilation calculated dynamically from formula

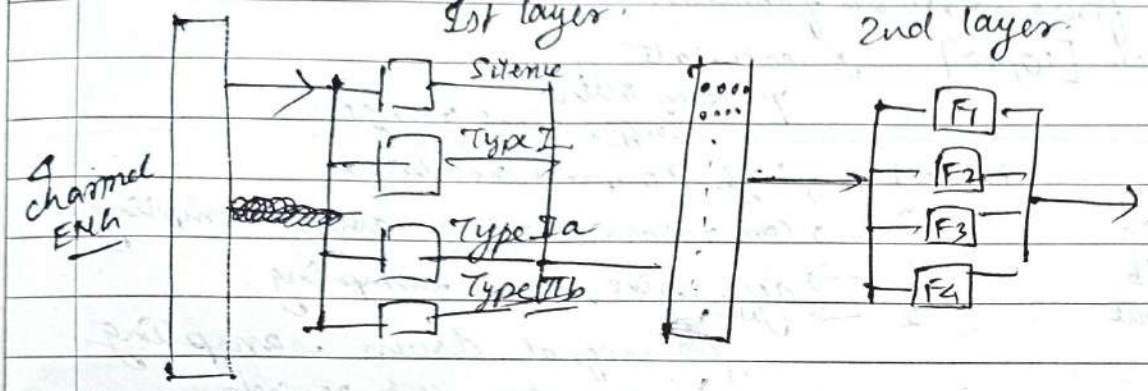
Type II b $k = 7$. Freq = 225 Hz
$L = 4$.
Stride = 1

=

Silence $k = 7$.
$L = 4$
Stride = 1.
no dilation

fixed feature
[-1 -1 -1 -1 -1 -1 -1]

$k = 28$



1D CNN is individually performed on each column ~~column~~ of each channel and output matrix is filled by the output columns of 2nd layer.

F1, F2, F3, F4 } identical 1D CNN. multilayered trainable features.

RELU present

If no stride.
Output = 4×4×4×n tensor.
i×j×k×n will represent: 64 data points at the sample 'n'

i = EMG signal
j = 1st layer feature
k → 2nd layer feature
n → sample

j=0 type I
=1 IIa
=2 IIb
=3. silence

for. i×0×k×n. S=1.
no stride

i×1×k×n S=2.

Total Receptive field

$$R = 1 + \sum_{i=1}^{L} (k_i - 1)(d_i)(S).$$

Teacher's Signature : _____

for 2nd layer

k = 10 fixed across all features.
dilations [10, 20] → to calculate patterns over longer time ranges

Type 1	stride = 1	→ high temporal resolution
Type IIa	2	→ can tolerate minor down sampling.
IIb	3	→ aggressive down sampling.
silence	1	→ no signal down sampling for rest periods.

MLP

Input layer = output from 2nd CNN.

Hidden layers → 128 with RELU Activation
→ 64 with RELU activation.

Output layer = 5 neurons,
each corresponding to 1 servo motor.
Predictions of servo motor angles from 0 to 180

Processing pipeline

Flattening.

output of 2nd CNN → 2D vector.

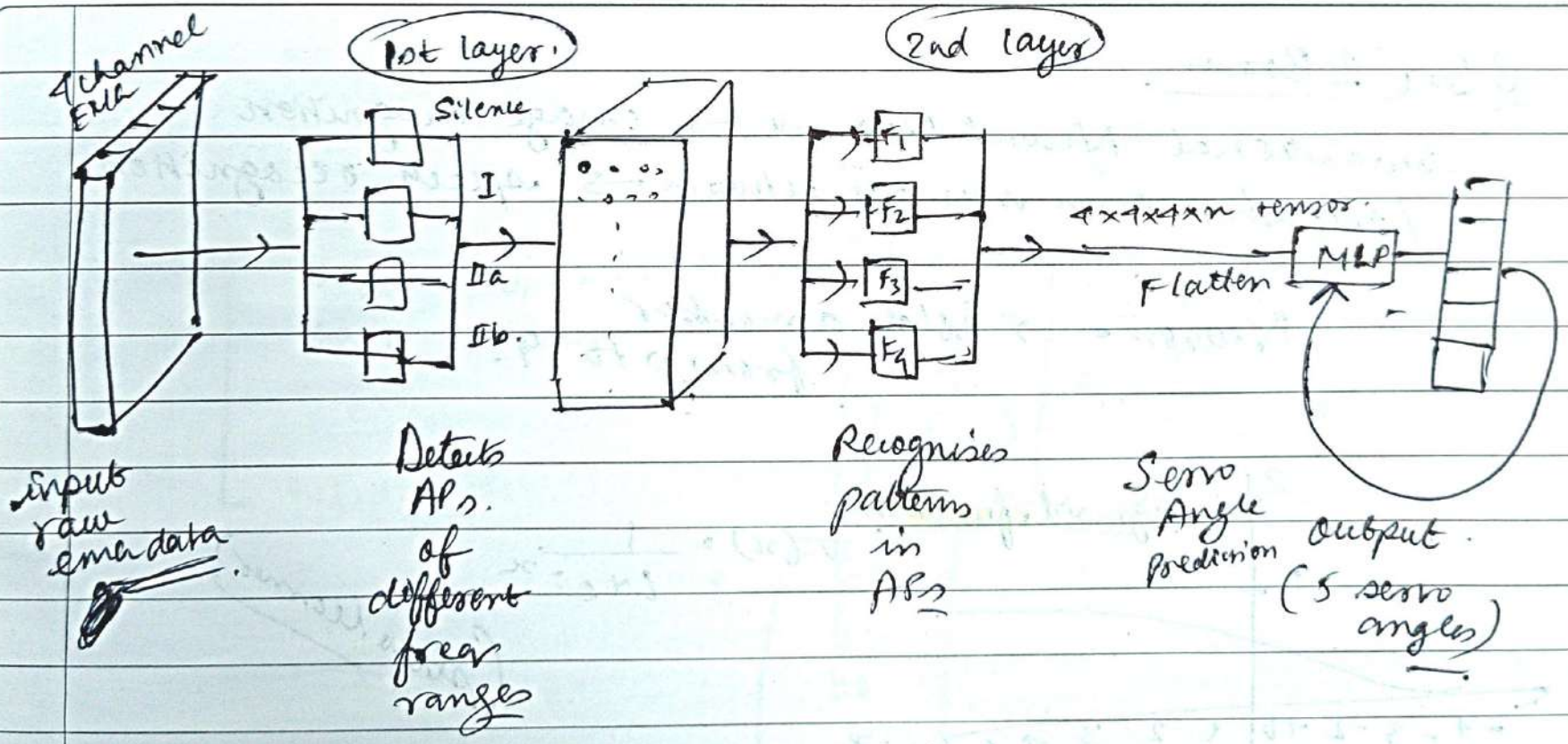
(4, 1, 4, 50) = 4 × 1 × 4 × 50

Concatenation

flattened feature vector is concatenated with historical servo angle data (A_{n}) and (A n-1) adding 10 neurons

Date

Expt. No. Page No.

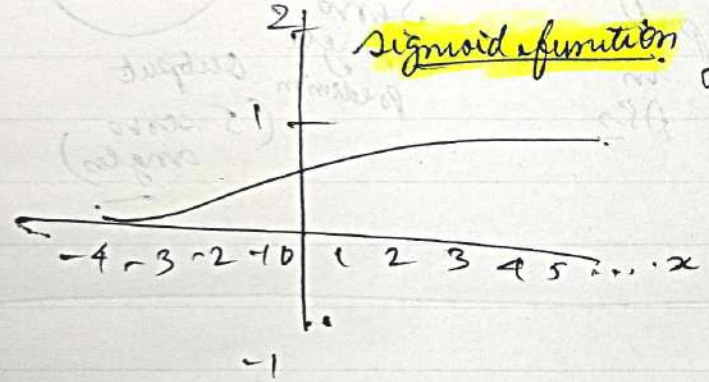


CNN Notes

3 Blue 1 Brown

Convolutional Neural Network → image recognition
- Long short term memory network → speech recognition.

Neuron ⟶ holds a number from 0 to 9.

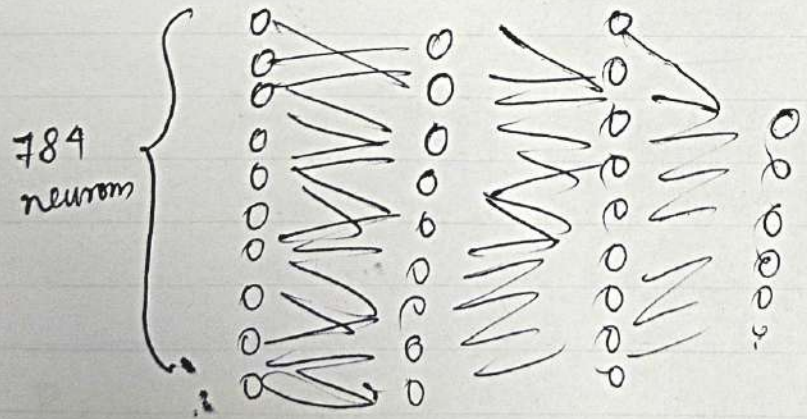


Sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$

(slow learner)

$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \dots + w_na_n - 10)$

10 ↓ bias



784 × 16 weights
16 biases.

16 + 16 + 10 biases
784 × 16 + 16 × 16 + 10 × 16 weights.

Learning → Finding the right weights & biases.

13002

Date

Expt. No. Page No.

$$a_0^{(1)} = \sigma\left(w_{0,0}\, a_0^{(0)} + w_{0,1}\, a_1^{(0)} + \cdots + w_{0,n}\, a_n^{(0)} + b^0\right)$$

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & & & \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} = \begin{bmatrix} \quad \\ \quad \\ \quad \\ \quad \end{bmatrix}$$

$$+ \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

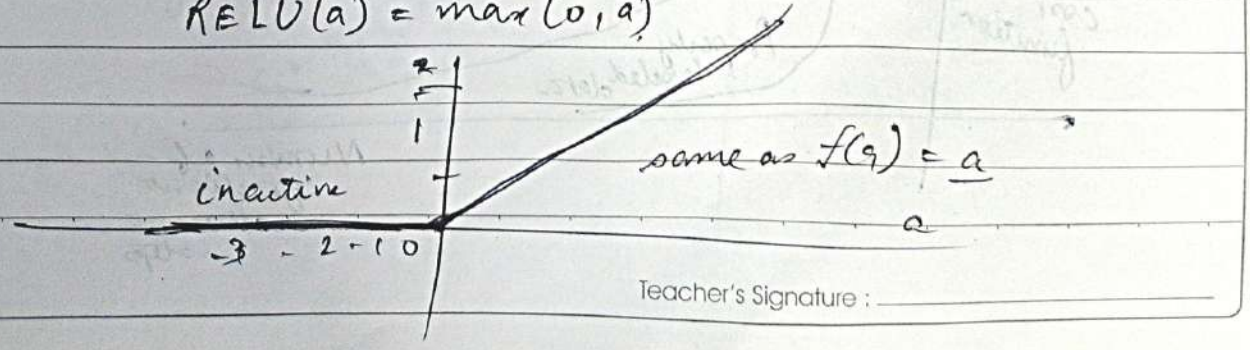
$$\sigma\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} \sigma(x) \\ \sigma(y) \\ \sigma(z) \end{bmatrix}$$

$$a^{(1)} = \sigma\left(W a^{(0)} + b\right)$$

Neuron → function.

Rectified linear Unit (RELU) (fast learner)

$$RELU(a) = \max(0, a)$$



Teacher's Signature :

Neural Network function

Input → 784 numbers / pixels.
Output → 10 numbers
parameters → 13002 weights / biases.

Cost function

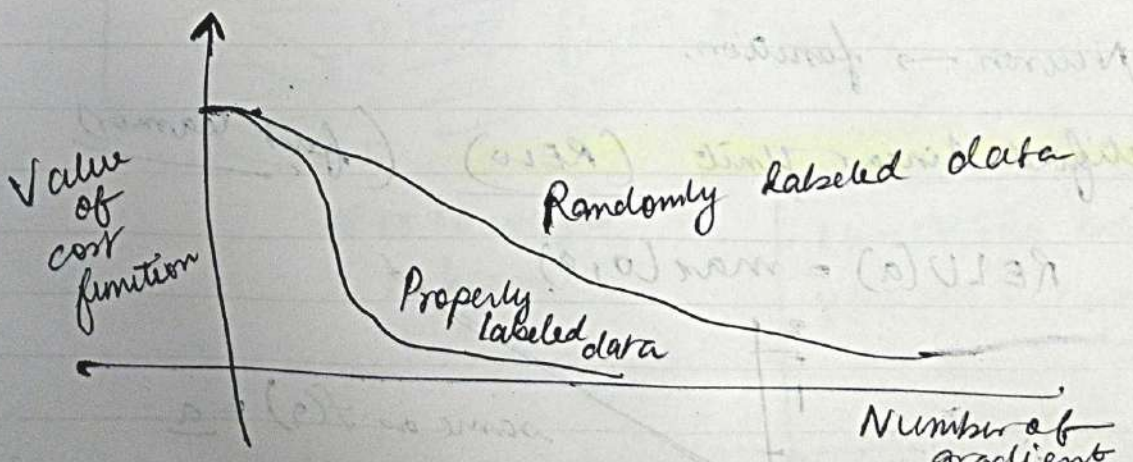
Input → 13002 weights / biases.
Output → 1 number
parameters → training examples

Gradient descent → to reach local minimum of cost function

$-\nabla C(\vec{w})$

Old technology → Multilayer Perceptron (MLP)

New → CNN
LSTM



Date

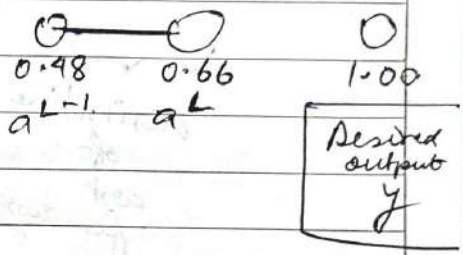
Expt. No. Page No.

Backpropagation

↓

a technique to understand how to modify weights and biases in training data to reduce cost function by gradient descent.

(MNIST Database)



Cost $\rightarrow C_0(\dots)$

$$C_0 = (a^L - y)^2$$

$$= (0.66 - 1)^2$$

$$a^L = \sigma(w^L a^{L-1} + b^L)$$

z^L

$$a^L = \sigma(z^L)$$

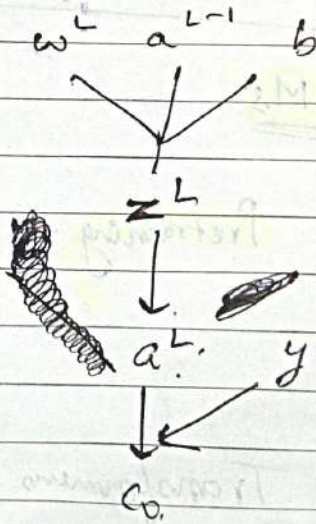
$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial C_0}{\partial a^L}$$

$$\frac{\partial C_0}{\partial a^L} = 2(a^L - y)$$

$$\frac{\partial a^L}{\partial z^L} = \sigma'(z^L)$$

$$\frac{\partial z^L}{\partial w^L} = a^{L-1}$$

$$\therefore \frac{\partial C_0}{\partial w^L} = (a^{L-1})(\sigma'(z^L)) \cdot 2(a^L - y)$$



Teacher's Signature :

$$\therefore \frac{\partial C}{\partial w^L} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial C_k}{\partial w^L}$$

derivative of cost function

Average of all training examples.

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^1} \\ \frac{\partial C}{\partial b^1} \\ \vdots \\ \frac{\partial C}{\partial w^L} \\ \frac{\partial C}{\partial b^L} \end{bmatrix}$$

LLMs

Pretraining → Reinforced Learning with Human Feedback

RLHF

Transformers use feedforward mechanism as well (attention)

Date

Expt. No. Page No.

GPT → generative pre-trained transformer

input → sentence/voice

↓

broken down into tokens/vectors (high dimensional)

GPT3 → first LLM model in world

total weights = 175,181,291,520

= 27938 matrices

Embedding matrix (W_E) → first layer

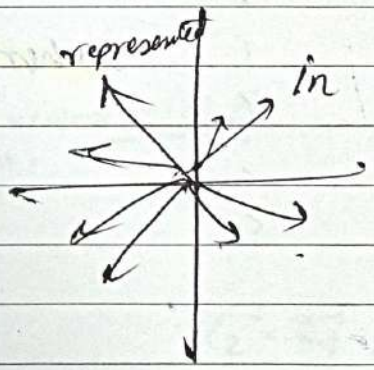
Words —embedding→ vectors.

Embedding space

↓

12288-dimensional

All data in deep learning must be represented as vectors



Unembedding matrix → last layer

$W_U \cdot \begin{bmatrix} \\ \end{bmatrix} = \begin{bmatrix} \\ \end{bmatrix}$ → softmax { probability distribution of next words

Teacher's Signature :

Softmax (technique to arrange numbers according to probability distribution from 0 to 1)

$e^{x_1} > 0$

$e^{x_2} > 0$

$\vdots$

$e^{x_n} > 0$

$= \sum_{n=0}^{N-1} e^{x_n}$

$e^{x_1} / \sum_{n=0}^{N-1} e^{x_n}$

$e^{x_2} / \sum_{n=0}^{N-1} e^{x_n}$

$\vdots$

with temperature

$e^{x_i/T} / \sum_{n=0}^{N-1} e^{x_n/T}$

Logits

when T is large, small values are given higher weight so that probability distribution becomes more uniform.

T is small: large values hold importance

Date

Expt. No. Page No.

a | fluffy | blue | creature | roamed | the | verdant | forest

$\vec{E_1}$ $\vec{E_2}$ . . . . . $\vec{E_8}$

$\vec{E_1}'$ $\vec{E_2}'$ . . . . . $\vec{E_8}'$

$\vec{E_1}$

↓ W_Q (query vector = 128 dimensional)

$\vec{Q_1}$

creature → $\vec{E_4}$ → ↓ W_Q → $\vec{Q_4}$

fluffy → $\vec{E_2}$ ⟷ $\vec{K_2}$ (W_K (key vector = 128 D))

$\vec{K_2} \cdot \vec{Q_4} \geq 0$ (related)

≤ 0 (unrelated)

$$\therefore \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{K^T Q}{\sqrt{d_K}}\right) V$$

Masking is a phenomenon by which $\vec{K} \cdot \vec{Q}$ values, being negative are treated as $-\infty$ so that tokens afterwards don't affect previous tokens!!

Teacher's Signature

Value matrix $\vec{E_1}$ $\vec{E_2}$ . . . .

$\vec{E_1}$. W_V . $\vec{V_1}$.

$\Delta\vec{E_1}$ $\Delta\vec{E_2}$ . . . .

$\vec{E_1}$
+
$\Delta\vec{E_1}$
||
$\vec{E_1}'$

One head of attention

value params
||
(# query params) + (# key params)

self attention → masking required
predict next words in a sentence

cross attention → no masking
audio to transcript
translation b/w 2 languages

Date

Expt. No. Page No.

Value vectors inside an attention head would have same dimension as the embedding space.
Involves taking weighted sums of large vectors, weighted by the attention pattern

Inside each head, use only the value down matrix to produce 128-D vectors.
Takes weighted sums of these with attention pattern.
Multiply each by heads value up matrix to get 12,288 dimensional vector that can be added to the embedding in that position.

Input → Attention → MLP → Attention → MLP.

↓

Output

MLP

non linear function

Linear → RELU → Linear → Output

First Name → Michael.

$$\begin{bmatrix} — \vec{R_0} — \\ — \vec{R_1} — \\ — \vec{R_2} — \\ \vdots \\ — \vec{R_n} — \end{bmatrix} \begin{bmatrix} \\ \vec{E} \\ \\ \end{bmatrix} = \begin{bmatrix} \vec{R_0} \cdot \vec{E} \\ \vec{R_1} \cdot \vec{E} \\ \vdots \\ \vec{R_n} \cdot \vec{E} \end{bmatrix} = [\text{redacted}]$$

→ { ≈ 1, if $\vec{E}$ encodes first Name Michael
≈ 0, if not }.

Bias

$$+ \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$$= (\vec{M} + \vec{J}) \cdot \vec{E} - 1$$

$$= \underbrace{\vec{M} \cdot \vec{E} + \vec{J} \cdot \vec{E}}$$

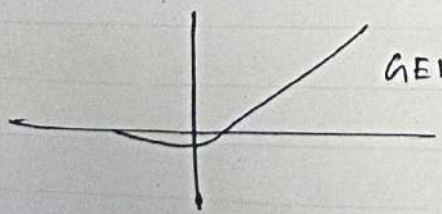
[redacted]

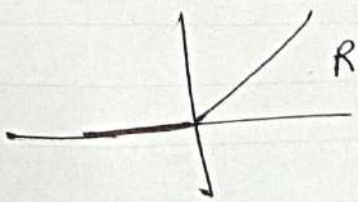
Upprojection

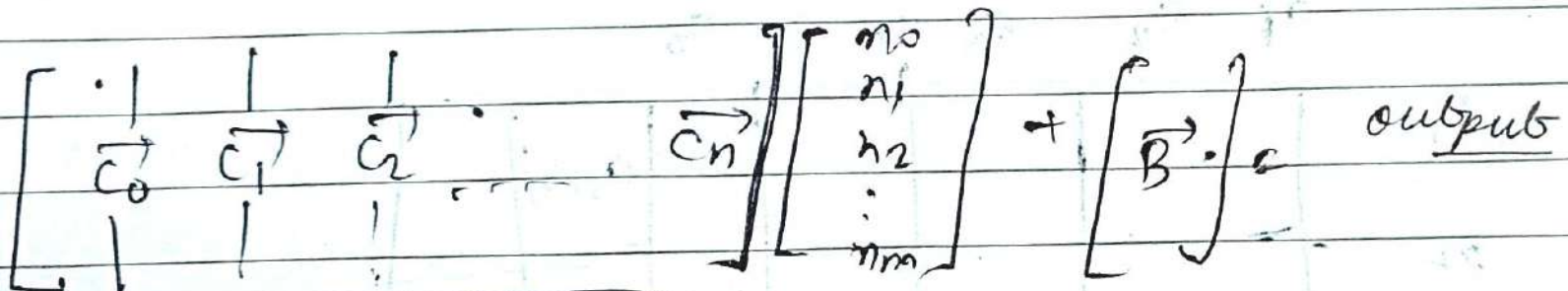
$$W_{\uparrow} \vec{E_i} + \vec{B_{\uparrow}} = \text{output matrix}$$

after passing through RELU, positive values are retained.
zero/negative values are made 0

same as ⇒ AND gate







$n_0 \vec{C_0} + n_1 \vec{C_1} + n_2 \vec{C_2} + \dots n_m \vec{C_m}$

down projection

$W\downarrow \square + \vec{B\downarrow} = \text{output}$

GPT 3

Embedding
Key
Query
Value

Output
Up projection.
Down projection

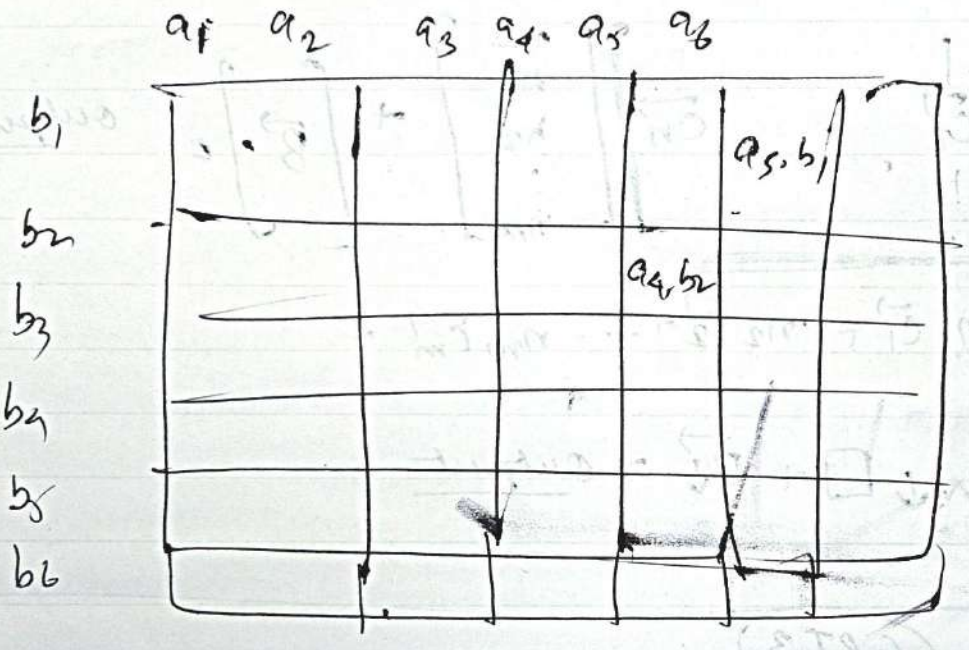
Unembedding

Johnson - Lindenstraus Lemma

⇒ · Maximum of vectors

$\approx \exp(\epsilon \cdot N)$

Convolution of (a_i) and (b_i) ⟶ Gaussian distribution



$$(a*b)_n = \sum_{\substack{i,j \\ i+j=n}} a_i \cdot b_j$$

→ n=0 to n=n

a_0	a_1	a_2
1	2	3

b_0	b_1	b_2
4	5	6

c_0 c_1 c_2 c_3 c_4

$= (4, 13, 28, 27, 18)$

(1.4) for n=0

$(1.5 + 2.4)$ for n=1

$(1.6 + 3\times4 + 2\times5)$ for n=2

$(2.6 + 3.5)$ for n=3

(3.6) for n=4

n=0 → $i_{min} + j_{min}$

n=4 → $i_{max} + j_{max}$

Date

Expt. No. Page No.

Fast convolution algorithm

$a = [a_0, a_1, a_2 \ldots\ldots a_{n-1}]$

$b = [b_0, b_1, \ldots\ldots b_{m-1}]$.

$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots\ldots a_{n-1} x^{n-1}$.

$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots\ldots g_{m-1} x^{m-1}$.

$a * b = f(x) \cdot g(x)$

$O(N^2)$ ✓

Const. N^2 + (stuff asymptotically smaller than N^2)

Fast fourier Transform (FFTs)

$\hat{a} = [\hat{a_0}, \hat{a_1}, \hat{a_2} \ldots\ldots \hat{a_{m+n-1}}]$

$\hat{b} = [\hat{b_0}, \hat{b_1}, \hat{b_2} \ldots\ldots \hat{b_{m+n-1}}]$

$\hat{a} \cdot \hat{b} = [\hat{a_0} \cdot \hat{b_0} + \ldots\ldots]$

↓ inverse FFT.

$a * b$

$O(N \log(N))$ ✓

Teacher's Signature :

CNN notes

DVLV

input dimensions are very big (for an MLP)

1 channel of an image 1920 × 1080 ≃ 2M features.

1 second of sound at 44kHz = 44K features.

So, a video → frame rate * image features + sound

10 seconds → (60 fps × 3 × 2M + 44K) * 10

music as input

↙ ↘ pleasant music / not. (1 → 100)

user preference ← regression

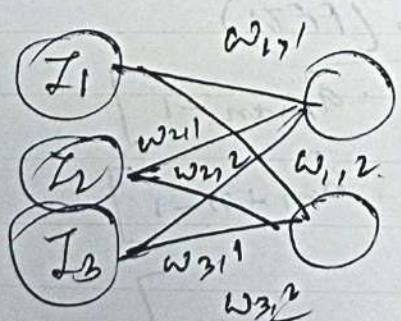
→ classifier

Traditional manual feature extraction

problem → Noise

Variations.

Fragments missing



$\sigma(\sum w_{i,1} * I_i)$

$\sigma(\sum w_{i,2} * I_i)$

MLP → ignores ordering of inputs.

Large MLP required with lots of weights.

Date

Expt. No. Page No.

We represent our filter as a vector which is equivalent to the weight matrix of an MLP. (bias)

The filter can act as a feature extractor.
filter can be used at all locations to detect visemes.
⟶ we call sliding a filter over all positions convolving
the operation is called a convolution operator.

The output of the convolution operator is itself again a vector.
A timeshift in the input causes the same shift in the output : Convolution is translation equivariant.

This operation can be understood as a small MLP that wanders across the input signal

In practice only, the conv kernels are learned.

Teacher's Signature :

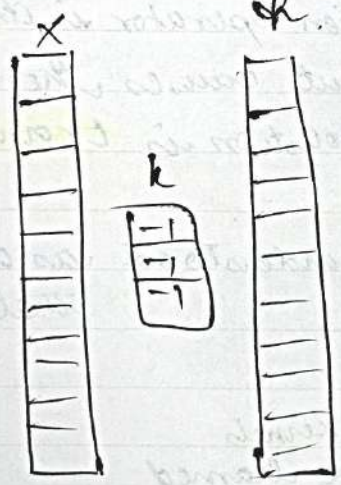
To understand how the conv kernels are learned, we require a formal definition.

For normal 1D input sequence $x \in R^n$

and a filter $\cdot K \in R^{2m+1}$

convolution operation is

$$h(t) = \cdot (x * K)(t) = \sum_{\tau=-m}^{m} x(t-\tau) \cdot K(\tau)$$



1DCNN

If k is learned, we update the filter weights based on some loss

$$L_h = \sum_{t=0}^{n} h_h(t)$$

depending on the conv. response at all places.

gradients utilized to update a kernel weights $K(\tau_0)$ is given by

eg. → cross correlation for classification

$$\frac{\partial L_h}{\partial K(\tau_0)} = \frac{\partial L_h}{\partial h} \cdot \frac{\partial h}{\partial K(\tau_0)}$$

$$= \sum_{t=0}^{n} \frac{\partial h_h(t)}{\partial h(t)} \sum_{\tau=-m}^{m} x(t-\tau) \cdot \frac{\partial K(\tau)}{\partial K(\tau_0)}$$

→ always zero except when $\tau = \tau_0$

Date

Expt. No. Page No.

The weights are shared across the entire input.
(MLPs learn independent weights at every position)

$h(t) = W(t, :) \cdot x(t)$.

Since weights are shared for every position,
Convolutional Neural Networks CNN are smaller than MLPs

Parameter efficiency

Convolutions can learn powerful pattern recognizers.
eg. for silence, based on "silences" appearing everywhere in the input.

(MLPs must learn an independent silence recognizer for every position)

Data efficiency

Convolutions can recognize id "silence pattern", regardless of where it appears.

(MLPs must have seen silence at a given position before in order to recognize it)

generalization improvement.

consequence of CNN being translation equivariant.

Teacher's Signature :

Near the boundaries of the data, we cannot apply the convolution as we normally do:-

Solutions

- Ignore boundaries.
- Pad boundaries
  - # → add (filter length − 1)/2 around the data to preserve the dimension.
    - → fill this with a fixed value, often 0.

we need to reduce dimension for the final classification.

- # → larger steps with our filter.
  - the size of the step is called the stride.

the dimension reduces with a factor equal to the stride.

- → input dimension must be a multiple of the stride.

Solutions → use a larger stride.
use a pooling layer
- → goes over the data similar to a convolution.
- → applies a deterministic function like max or average on the input.
- → usually has stride == pool size

Date

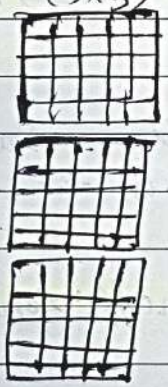
Expt. No. Page No.

2D-CNN

If we work with a 2D image → becomes a 3D tensor.
3D video → becomes a 4D tensor.

We have a 5×5 image.
3 channels.
2 filters → (3D)

(5×5)



x[1,1,0]
x[1,1,1]
x[1,1,2]

filter W0 (3×3×3)
W1 (3×3×3)

Bias. b0 (1×1×1) = 1
b1 (1×1×1) = 0

We add padding to solve issues with convolving near the borders.

∴ Input volume + (pad 1)
→ (7×7×3)

Output volume (3×3×2)

A translation of the input produces an equivalent translation in the output.

for a translated input $x(t - t_0)$:

$$(x * k)(t - t_0) = \sum_{\tau=-m}^{m} x((t - t_0) - \tau) \cdot k(\tau)$$

Teacher's Signature :

CNN is equivariant to translation
non-equivariant to rotation/scale mapping.

Convolution - Equivariance analysis

Problem!- Each of these filters are independent weights:-

The network wastes a lot of parameters learning transformed versions of the same
→ Parameter inefficient!!

To learn these filters, the network must see these transformations in the training set
→ Data inefficient + No equivariance guarantees

Solutions ?? (Group convolutions)

→ not just share parameters for translation but also other transformations!!

→ Extreme parameters sharing + equivariance guarantees.
→ active field of research.

Date.....

Expt. No.---------- Page No.----------

Recurrent architectures eg.- RNNs, LSTMs, ...
are able to handle arbitrarily long sequences.
→ via recurrence.

However, suffer from vanishing / exploding gradients problem.
(difficult to train and to learn from past)

CNNs offer interesting alternative for sequence modelling

Casuality is easily obtained by padding asymmetrically for a convolutional kernel of size K.
→ add padding "K-1"

→ the output at position "t" can be dependent on input values upto "K-1" steps in the past.
"The space it sees is called receptive field"

→ dilation factor d

We can stack several convolutional layers to form dilated causal convolutional networks.
→ Temporal Convolutional Networks (TCNs)

TCN's do not have recurrent connections,
they do not use Back-Propagation through time,
→ Trained in parallel.
Much faster training + Optimal GPU usage.

Teacher's Signature : _____

→ TCNs do not exhibit exploding/vanishing gradient descent problems along time axis

→ They learn from past by input values within their receptive fields.

(Input values outside receptive field cannot be considered for the calculation of output at a particular position)

In order to avoid vanishing gradients and improve learning, TCNs use batch normalization residual connections and dropout.

Residual block (k, d)