BASICS OF NEURAL NETWORKS

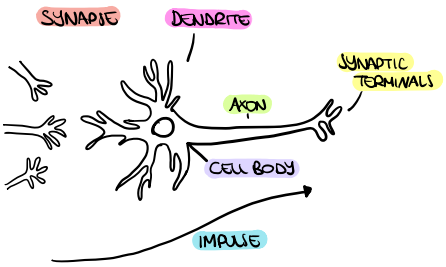
DEEP LEARNING IS EVERYWHERE → VISION, LANGUAGE, SPEECH

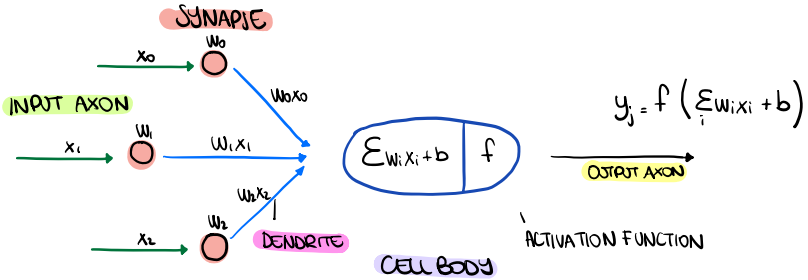
THE PERFORMANCE FOR DEEP LEARNING IS IMPROVING VERY RAPIDLY.

TERMINOLOGY OF NEURAL NETWORKS

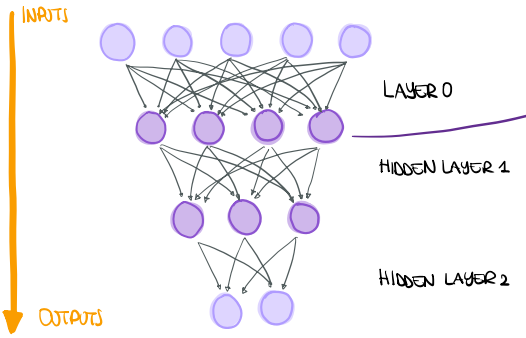
NEURON AND SYNAPSE

A SYNAPSE HAS MANY INPUT AXONS AND THEN WE MULTIPLY THE WEIGHTS OF THE INPUTS AXONS AND COMBINE THEM TOGETHER.





EXAMPLE OF A 3-LAYER NEURAL NETWORK



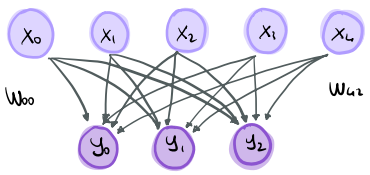
NOTE ON TERMINOLOGY:

SYNAPSES = WEIGHTS = PARAMETERS

NEURONS = FEATURES = ACTIVATIONS

THE DIMENSIONALITY OF THESE HIDDEN LAYERS DETERMINES THE WIDTH OF THE MODEL

THIS NETWORK HAS FULLY-CONNECTED LAYERS (LINEAR LAYERS) → THE OUTPUT NEURON IS CONNECTED TO ALL INPUTS NEURONS.



$$y_i = \sum_j w_{ij} x_j + b_i$$

SHAPE OF TENSORS:

INPUT FEATURES X: $(1, C_i)$

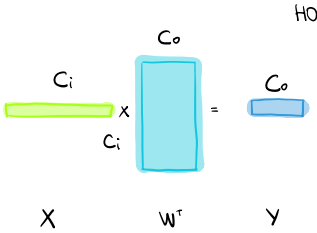
OUTPUT FEATURES Y: $(1, C_o)$

WEIGHTS W: (C_o, C_i)

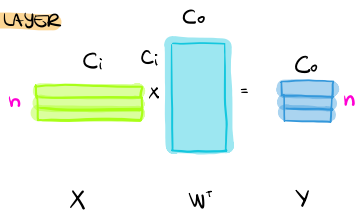
BIAS b: $(C_o,)$

WHERE C_i ARE THE INPUT CHANNELS AND C_o THE OUTPUT CHANNELS.

HOW TO CALCULATE THE DIMENSION OF A MULTILAYER PERCEPTRON:



IF WE HAVE A BATCH SIZE OF n →



SHAPE OF TENSORS:

INPUT FEATURES X: (n, C_i)

OUTPUT FEATURES Y: (n, C_o)

WEIGHTS W: (C_o, C_i)

BIAS b: $(C_o,)$

LET'S NOW LOOK AT A SLIGHTLY MORE COMPLICATED SCENARIO → CONVOLUTION LAYER

↓

THE OUTPUT NEURON IS CONNECTED TO INPUT NEURONS IN THE RECEPTIVE FIELD

↓

RATHER THAN HAVING JUST ONE CHANNEL, WE HAVE MULTIPLE DIMENSIONS IN THIS SPATIAL DIMENSION.

MANY SIGNALS ARE SIMILAR TO THIS CASE E.G: SPEECH SIGNALS

↳ FOR SPEECH RECOGNITION WE HAVE A LONG TEMPORAL SEQUENCE

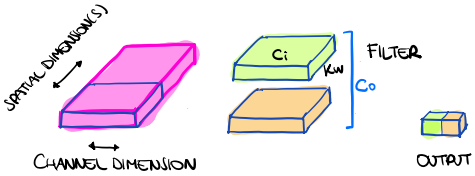
SHAPE OF TENSORS: 1D CONV

INPUT FEATURES X: (n, C_i, W_i) → INPUT/OUTPUT WIDTH

OUTPUT FEATURES Y: (n, C_o, W_o)

WEIGHTS W: (C_o, C_i, K_w) → KERNEL WIDTH

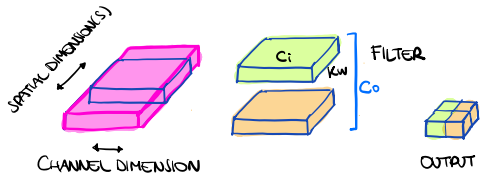
BIAS b: $(C_o,)$



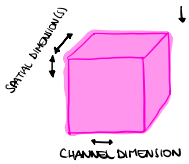
THE ORANGE FILTER PRODUCES THE ORANGE OUTPUT AND THE GREEN FILTER PRODUCES THE GREEN OUTPUT.

THE ONE DIMENSIONAL CONVOLUTION IS COMPUTED BY PERFORMING THE MULTIPLICATION OF EACH ELEMENT AND SUM THEM UP TOGETHER TO PRODUCE ONE OUTPUT ELEMENT. THERE ARE MULTIPLE FILTERS ↳ WHICH CAN PRODUCE ANOTHER CHANNEL IN THE SAME DIMENSION.

IN ORDER TO PRODUCE THE NEXT SPATIAL DIMENSION WE CAN SHIFT THE INPUT BY ONE → SO WE CAN PRODUCE THE NEXT SPATIAL OUTPUT.

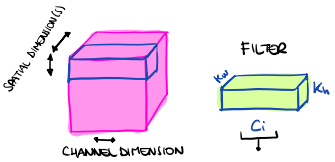


LET'S MOVE TO A MORE COMPLICATED SCENARIO → 2D CONVOLUTION → WIDELY USED IN PROCESSING IMAGES



SHAPE OF TENSORS:	1D CONV	2D CONV (INPUT / OUTPUT HEIGHT)
INPUT FEATURES X:	(n, C_i, W_i)	(n, C_i, h_i, W_i)
OUTPUT FEATURES Y:	(n, C_o, W_o)	(n, C_o, h_o, W_o)
WEIGHTS W:	(C_o, C_i, K_w)	(C_o, C_i, K_h, K_w)
BIAS b:	$(C_o,)$	$(C_o,)$

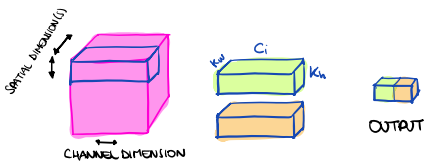
IN ORDER TO PRODUCE THE OUTPUT WE USE AGAIN A FILTER → THE DIMENSION OF THE FILTER IS DENOTED BY THE K, WHICH IS INDICATING "KERNEL".



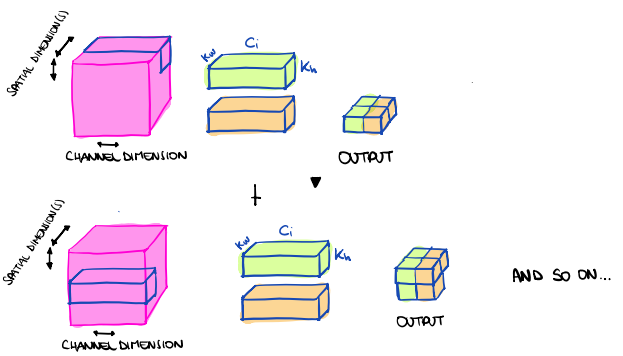
AFTER THE DOT PRODUCT WE PRODUCE JUST ONE OUTPUT ELEMENT → SO FOR ONE SINGLE OUTPUT ELEMENT WE NEED: $C_i \times K_h \times K_w$ MULTIPLICATIONS

THE CHANNEL SHOULD BE EXACTLY THE SAME AS THE INPUT → SO WE CAN DO THE DOT PRODUCT WITH A PORTION OF THE INPUT.

IF WE WANT TO HAVE MULTIPLE OUTPUT CHANNELS WE ARE GOING TO HAVE MULTIPLE FILTERS:



AND THEN WE ARE GOING TO MOVE THE INPUT IN THE SPATIAL DIMENSION TO CALCULATE THE OTHER SPATIAL DIMENSIONS OF THE OUTPUT.



HOW DO WE CALCULATE THE OUTPUT DIMENSION?

→ OUTPUT FEATURES Y → (n, C_o, h_o, W_o)

WHERE $h_o = h_i - K_h + 1$

PADDING → PADDING CAN BE USED TO KEEP THE OUTPUT FEATURE MAP SIZE THE SAME AS THE INPUT FEATURE MAP SIZE.

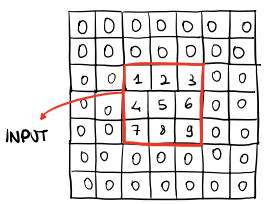
THERE ARE SEVERAL APPROACHES TO DO THE PADDING, THE SIMPLEST ONE IS TO DO **ZERO PADDING**.

→ WITH PADDING THE OUTPUT SIZE IS CALCULATED USING: $h_o = h_i + 2p - K_h + 1$

→ WHERE p IS THE PADDING SIZE

OTHER PADDING:

- **REFLECTION PADDING** → IT "REFLECTS" THE ROW INTO THE PADDING
- **REPLICATION PADDING** → REPLICATION IS DONE BY COPYING THE NEAREST BORDER VALUE



IT PADS THE INPUT BOUNDARIES WITH ZERO (IT'S THE DEFAULT IN PYTORCH)

CONVOLUTION LAYER → RECEPTIVE FIELD → WE WANT TO HAVE A GLOBAL UNDERSTANDING OF THE WHOLE IMAGE TO UNDERSTAND WHAT ARE THE HIGH LEVEL FEATURES.

→ IN CONVOLUTION EACH OUTPUT ELEMENT DEPENDS ON $K_h \times K_w$ RECEPTIVE FIELD IN THE INPUT.

- EACH SUCCESSIVE CONVOLUTION ADDS K-1 TO THE RECEPTIVE FIELD SIZE
- WITH L LAYERS, THE RECEPTIVE FIELD SIZE IS $L \cdot (K-1) + 1$

→ THE **PROBLEM** IS THAT INORDER TO HAVE A LARGE RECEPTIVE FIELD WE NEED MANY LAYERS FOR EACH OUTPUT TO "SEE" THE WHOLE IMAGE.

→ **SOLUTION**: DOWNSAMPLE INSIDE THE NEURAL NETWORK

THE STRIDE CONVOLUTION LAYER INCREASES THE RECEPTIVE FIELD.

STRIDED CONVOLUTION LAYER → $h_o = \frac{h_i + 2p - K_h}{S} + 1$ WHERE P IS PADDING AND S IS STRIDE.

GROUPED CONVOLUTION LAYER → A METHOD TO REDUCE THE AMOUNT OF WEIGHTS → A GROUP OF NARROWER CONVOLUTIONS

SHAPE OF TENSORS	
INPUT FEATURES X:	(n, C_i, h_i, w_i)
OUTPUT FEATURES Y:	(n, C_o, h_o, W_o)
WEIGHTS W:	$(g \cdot C_o/g, C_i/g, K_h, K_w)$ (g = GROUPS)
BIAS b	$(C_o,)$

PREVIOUSLY EACH OUTPUT CHANNEL WAS CALCULATED FOR EACH INPUT CHANNEL, IN A GROUPED CONVOLUTION WE HAVE DIFFERENT GROUPS, EACH GROUP ONLY DEPENDS ON A SPECIFIC GROUP OF INPUT RATHER THAN THE ENTIRE INPUT CHANNELS.

DEPTHWISE CONVOLUTION LAYER → THAT IS AN EXTREME CASE OF GROUPED CONVOLUTION WHERE THE NUMBER OF GROUPS IS EQUAL TO THE NUMBER OF CHANNELS.

INDEPENDENT FILTER FOR EACH CHANNEL → $g = C_i = C_o$ IN GROUPED CONVOLUTION.

THIS GREATLY SIMPLIFIES THE CONVOLUTION AND THE SIZE OF THE CONVOLUTION WEIGHT AND ALSO THE NUMBER OF OPERATIONS.

SHAPE OF TENSORS

INPUT FEATURES X: (n, C_i, h_i, w_i)

OUTPUT FEATURES Y: (n, C_o, h_o, w_o)

WEIGHTS W: (C, K_h, K_w)

BIAS b: $(C_o,)$

$$h_o = \frac{h_i + 2p - K_h}{s} + 1$$

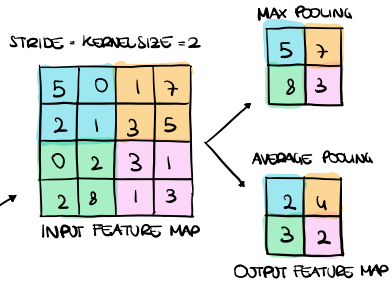
MORE FANCIER OPERATIONS:

DILATED CONVOLUTION TRANSPOSED CONVOLUTION ETC..

↳ USED FOR PIXEL-WISE PREDICTIONS

WHEN WE WANT TO DOWNSAMPLE THE FEATURE MAP TO A SMALLER SIZE

POOLING LAYER



THE OUTPUT NEURON POOLS THE FEATURES IN THE RECEPTIVE FIELD

USUALLY THE STRIDE IS THE SAME AS THE KERNEL SIZE: $s = K$

POOLING OPERATES OVER EACH CHANNEL INDEPENDENTLY

NO LEARNABLE PARAMETERS

NORMALIZATION LAYER

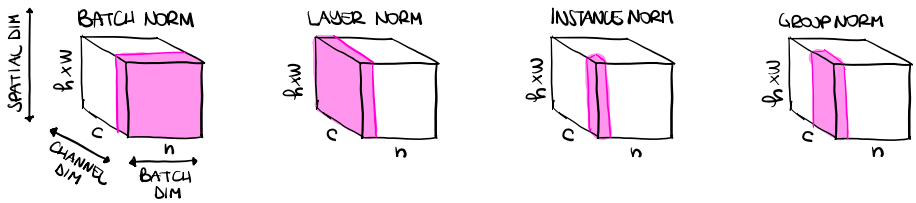
NORMALIZING THE FEATURES MAKES TRAINING FASTER

NORMALIZATION LAYER NORMALIZES THE FEATURES AS FOLLOWS → $\hat{x}_i = \frac{1}{\sigma}(x_i - \mu)$ WHERE $\mu_i = \frac{1}{m}\sum_{k \in \delta_i} x_k$ AND $\sigma_i = \sqrt{\frac{1}{m}\sum_{k \in \delta_i}(x_k - \mu_i)^2 + \varepsilon}$ ARE THE MEAN AND STANDARD DEVIATION OVER THE SET OF PIXELS δ_i

↓

THEN LEARNS A PER-CHANNEL LINEAR TRANSFORM (TRAINABLE SCALE γ AND SHIFT β) TO COMPENSATE FOR THE POSSIBLE LOST OF REPRESENTATIONAL ABILITY → $y = \gamma_{i_c}\hat{x}_i + \beta_{i_c}$

NOTE: DIFFERENT NORMALIZATIONS USE DIFFERENT DEFINITIONS OF THE SET δ_i COLORED IN PINK



SOMETIMES THESE NORMALIZATION LAYERS CAN IMPROVE THE TRAINING CONVERGENCE BUT SOMETIMES THEY MAKE INFERENCE MORE CHALLENGING.

IN EFFICIENT NEURAL NETWORKS SOME OF THE NORMALIZATION LAYERS CAN BE FOLDED INTO A CONVOLUTION LAYER TO SAVE THE COMPUTATION!

ACTIVATION FUNCTION

ACTIVATION FUNCTIONS ARE TIPICALLY NON LINEAR FUNCTIONS

SIGMOID $y = \frac{1}{1+e^{-x}}$

RELU $y = \max(0, x)$ ← ONE OF THE MOST EFFICIENT FROM THE IMPLEMENTATION PERSPECTIVE SINCE IT DOESN'T REQUIRE COMPLICATED ARITHMETIC

RELU6 $\min(\max(0, x), 6)$ → AND IT'S VERY QUANTIZATION FRIENDLY! → WE CAN THRESHOLD THE OUTPUT TO BE SOME NUMBER.

LEAKY RELU $y = \max(ax, x)$

SWISH $y = \frac{x}{(1+e^{-x})}$ ← IT HELPS TO HAVE A BOOST IN THE ACCURACY BUT NOT VERY HARDWARE FRIENDLY FOR ITS IMPLEMENTATION; NOT VERY QUANTIZATION FRIENDLY

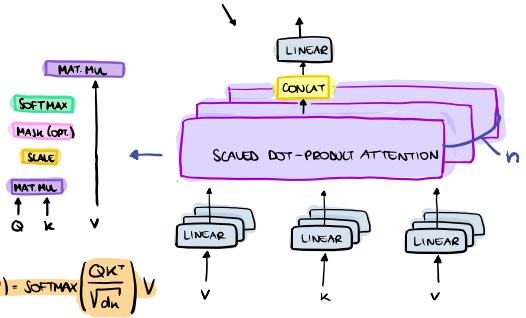
HARD SWISH $y = x \cdot \frac{\text{RELU6}(x+3)}{6}$

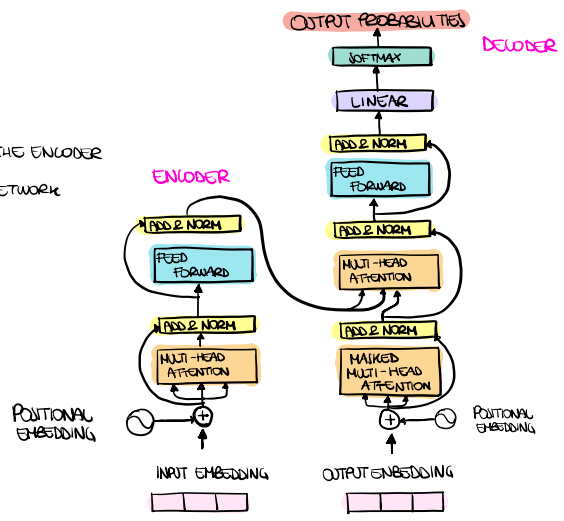
TRANSFORMER THE TRANSFORMER HAS TWO PARTS: THE ENCODER AND THE DECODER. FOR THE ENCODER WE HAVE TWO PARTS ONE IS THE MULTI HEAD ATTENTION, FOLLOWED BY THE FEED FORWARD NETWORK

WE HAVE THREE INPUTS: THE VALUE, THE QUERY AND THE KEY. EACH OF THEM PASSES THROUGH A LINEAR TRANSFORMATION AND THEN WE DO THE ATTENTION MECHANISM

IT TRIES TO FIND THE RELATIONSHIP BETWEEN EACH TOKEN

$$\text{ATTENTION}(Q, K, V) = \text{SOFTMAX}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$





PROBLEM IF YOU HAVE A LONG SEQUENCE (WHERE n IS LARGE) IS A PROBLEM BECAUSE THE ATTENTION MAP IS n^2 ← THE COMPLEXITY GROWS QUADRATICALLY WITH THE SEQUENCE LENGTH

POPULAR CNN ARCHITECTURES

ALEXNET

- 5 CONVOLUTION LAYERS
- 3 FULLY CONNECTED LAYERS

AFTER EACH STAGE WITH A STRIDE OF TWO WE HAVE A RESOLUTION ROUGHLY HALF OF THE PREVIOUS STAGE

VGG-16

IT HAS 16 LAYERS. EACH LAYER CONTAINS THE CONVOLUTION BATCH NORMALIZATION AND THE RELU

RESNET-50

IT INTRODUCED THE RESIDUAL BRANCH SO THAT WE CAN MAKE THE FLOW EASIER AND ENABLE A DEEPER NEURAL NET.

MOBILE NETV2

USED FOR MOBILE APPLICATIONS

↑ STEP-WISE CONVOLUTION

EFFICIENCY METRICS → HOW TO MEASURE THE EFFICIENCY OF DIFFERENT NEURAL NETS

↳ FOR A NEURAL NET WE WANT IT TO BE SMALLER, FASTER AND GREENER → ENERGY CONSUMPTION

- **STORAGE**: HOW MANY STORAGE DO YOU NEED TO STORE YOUR NN
- **LATENCY**: HOW FAST CAN YOU PROCESS FROM THE INPUT TO THE OUTPUT

ALL THESE CONCERN TWO FACTORS OF A NEURAL NETWORK: COMPUTATION AND MEMORY

NOTE:

MEMORY ≠ STORAGE. STORAGE IS A STATIC CONCEPT OF ONLY THE WEIGHTS AND CODE, MEMORY ALSO INCLUDES THE ACTIVATIONS.

THE EFFICIENCY METRICS ARE COVERED FROM TWO PERSPECTIVES → COMPUTATION RELATED → · MACs · FLOP

MEMORY RELATED

MODEL SIZE ← · NUMBER OF PARAMETERS

· THE TOTAL ACTIVATIONS AND THE PEAK ACTIVATIONS

INPUT ACTIVATION SIZE + OUTPUT ACTIVATION SIZE → NN SPECIFICATION

MEMORY BANDWITH OF PROCESSOR → HARDWARE SPECIFICATION

HOW TO CALCULATE THE LATENCY → $\doteq \max(T_{COMPUTATION}, T_{MEMORY}) \doteq T_{DATA\ MOVEMENT\ OF\ ACTIVATIONS} + T_{DATA\ MOVEMENT\ OF\ WEIGHTS}$

$\doteq$ NUMBER OF OPERATIONS IN NEURAL NETWORKS MODEL → NN SPECIFICATION

HARDWARE SPECIFICATION ← NUMBER OF OPERATIONS THAT PROCESSOR CAN PROCESS PER SECONDS

$\frac{\text{NEURAL NETWORK MODEL SIZE}}{\text{MEMORY BANDWITH OF PROCESSOR}}$ → NN SPECIFICATION / HARDWARE SPECIFICATION

ENERGY CONSUMPTION → MEMORY MOVEMENT IS MUCH MORE EXPENSIVE THAN COMPUTATION! ⇒ WE WANT TO REDUCE THE AMOUNT OF DATA MOVEMENT WHEN WE ARE DESIGNING NEURAL NETS.

 # OF PARAMETERS → IT'S THE NUMBER OF WEIGHTS FOR A GIVEN NEURAL NETWORK

MODEL SIZE → · IT MEASURES THE STORAGE FOR THE WEIGHTS OF A GIVEN NN.

- THE COMMON UNITS FOR THE MODEL SIZE ARE MEGABYTE (MB), KILOBYTE (KB) AND BITS.
- IN GENERAL (IF THE WHOLE NN USES THE SAME DATA TYPE: MODEL SIZE = #PARAMETERS × BIT WIDTH
  - ↳ WE CAN USE 32-BIT FLOATING POINT OR 16 ETC..

 # OF ACTIVATIONS → IS THE MEMORY BOTTLENECK IN INFERENCE ON IOT

THE DISTRIBUTION OF THE ACTIVATION IS HIGHLY IMBALANCED → SOME OF THE LAYERS CAN HAVE A DOMINATING ACTIVATION → SO WHAT IS THE LEAST AMOUNT OF MEMORY TO PROCESS THIS NN? THAT IS IMPACTED BY THE PEAK FOR INFERENCE, FOR TRAINING THAT'S IMPACT BY THE SUM.

↑ FOR TRAINING THE ACTIVATION IS THE BOTTLENECK (NOT # OF PARAMETERS)

MACs → NUMBER OF MULTIPLY-ACCUMULATE OPERATIONS

LAYER	MACs (BATCH SIZE = 1)
LINEAR	$C_o \cdot C_i$
CONVOLUTION	$C_i \cdot K_h \cdot K_w \cdot h_o \cdot W_o \cdot C_o$
GROUPED CONVOLUTION	$C_i/g \cdot K_h \cdot K_w \cdot h_o \cdot W_o \cdot C_o$
DEPTH WISE CONVOLUTION	$K_h \cdot K_w \cdot h_o \cdot W_o \cdot C_o$

FLOP → # OF FLOATING POINT OPERATIONS

↳ A MULTIPLY IS A FLOATING POINT OPERATION AND ALSO AN ADD.

· ONE MULTIPLY-ACCUMULATION OPERATION IS TWO FLOATING POINT OPERATIONS.