σε ένα cnn με ποιά σειρά πάνε τα layers ;;

πάει έτσι: convulution layer, ReLU layer, Max pooling και τέλος batch normalization

Η συνηθισμένη σειρά των layers σε ένα Convolutional Neural Network (CNN) **δεν είναι ακριβώς** όπως την έγραψες, αλλά σχεδόν. Η **τυπική σειρά** είναι:

1. Convolutional layer

2. Batch Normalization (προαιρετικά, αλλά συχνά ακολουθεί το conv)

3. Activation function (π.χ. ReLU)

4. Pooling layer (π.χ. Max Pooling)

Δηλαδή:

Conv → BatchNorm → ReLU → MaxPool

**Αναλυτικά:**

* **Convolutional Layer**: Εφαρμόζει φίλτρα (kernels) για εξαγωγή χαρακτηριστικών (features).
* **Batch Normalization**: Ομαλοποιεί την έξοδο του conv layer (για σταθερότερη και ταχύτερη εκπαίδευση).
* **ReLU (ή άλλη activation)**: Εισάγει μη γραμμικότητα στο μοντέλο.
* **Max Pooling (ή Average Pooling)**: Μειώνει τη χωρική διάσταση (downsampling) και βοηθά στη μείωση του overfitting.

**Εναλλακτικές διατάξεις**

Κάποιες αρχιτεκτονικές βάζουν το **ReLU πριν ή μετά το BatchNorm**, αλλά πιο συχνή και αποδοτική στην πράξη είναι η:

Conv → BatchNorm → ReLU

(κι αν υπάρχει pooling, μετά από αυτά.)

**Παράδειγμα**

nn.Sequential(

nn.Conv2d(in\_channels, out\_channels, kernel\_size),

nn.BatchNorm2d(out\_channels),

nn.ReLU(),

nn.MaxPool2d(kernel\_size=2)

)

γιατί κάνουμε normalization πριν αφαιρέσουμε τις αρνητικές τιμές με την ReLU;;

Δεν θα ήταν πιο έυκολο να κάνουμε normalization με μόνο θετικές τιμές ή μηδέν;;

Πολύ ωραία ερώτηση! Είναι εύλογο να σκέφτεσαι ότι θα ήταν πιο "εύκολο" να κάνεις **Batch Normalization (BN)** αφού έχουν αφαιρεθεί οι αρνητικές τιμές με το **ReLU**. Όμως στην πράξη κάνουμε **BatchNorm πριν τη ReLU** για πολύ συγκεκριμένους και σημαντικούς λόγους.

Ας το εξηγήσουμε βήμα-βήμα.

**🔁 Τι κάνει η Batch Normalization;**

Η **BatchNorm** παίρνει την έξοδο ενός layer (π.χ. convolutional) και την:

1. **Ομαλοποιεί** ώστε να έχει μέσο όρο ≈ 0 και τυπική απόκλιση ≈ 1, σε κάθε mini-batch.
2. Μετά εφαρμόζει ένα γραμμικό transformation:

A black screen with white text

AI-generated content may be incorrect.όπου γ\gamma και β\beta είναι trainable παράμετροι.

Άρα, η έξοδος της BN μπορεί να έχει **αρνητικές και θετικές τιμές**.

**🔍 Γιατί έρχεται πριν από τη ReLU;**

Αν έκανες το ReLU **πριν** τη BN, τότε:

* Οι τιμές που πάνε στη BN θα ήταν **μόνο μη αρνητικές**.
* Δηλαδή ολόκληρη η κατανομή της εξόδου θα ήταν skewed — με **μέσο όρο > 0**, και **όχι καθόλου συμμετρική**.
* Η BN σε αυτή την περίπτωση **δεν θα δούλευε σωστά**, γιατί δεν θα μπορούσε να ομαλοποιήσει αποτελεσματικά.
* Θα είχες πρόβλημα και με το training, γιατί τα gradients θα ήταν πιο ασταθή.

Αντίθετα, όταν βάζεις BN **πριν** από ReLU:

* Η BN βλέπει **ολόκληρη την κατανομή** των τιμών (αρνητικές & θετικές).
* Κάνει τη δουλειά της πιο σωστά (μηδενίζει το μέσο όρο, τυποποιεί).
* Μετά έρχεται η ReLU και μηδενίζει τα αρνητικά.

**✅ Γενικός κανόνας**

**Always apply BatchNorm before ReLU (or other non-linear activation).**

Αυτό ισχύει και στη βιβλιογραφία και σε σχεδόν όλες τις επιτυχημένες αρχιτεκτονικές όπως ResNet, VGG με BN, κ.λπ.

**Αντίστροφη σειρά: Πότε (αν ποτέ) έχει νόημα;**

* Σπάνια, πειραματικά σε κάποιες αρχιτεκτονικές (π.χ. LayerNorm σε NLP) μπορεί να μπει activation πρώτα.
* Αλλά με BN σε CNNs, η **σωστή σειρά είναι: Conv → BN → ReLU**

**Θες παράδειγμα με γραφήματα ή να το δούμε με PyTorch σε πράξη;**