ΕΘΝΙΚΟ ΚΑΙ
ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

2η Εργασία

Ιωάννης Δαλιάνης 1115201700027

Γεώργιος Φλώρος 1115201700178

Τμήμα Πληροφορικής και Τηλεπικοινωνιών, ΕΚΠΑ

Νοέμβριος 2020

Περιεχόμενα

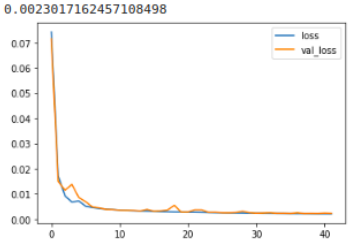
1 Α Μέρος

Το πρόγραμμα μας δημιουργεί ένα μοντέλο δυναμικά,ανάλογα με το input και τις υπερπαραμέτρους που καθορίζει ο χρήστης. Στο πρόγραμμα μας ορίζουμε ως block έναν συνδιασμό Convolution - Batch Normalization - Max Pooling. Σε ένα block μπορούν να γίνουν όσα convolutions-batch normalizations επιθυμεί ο χρήστης και στην συνέχεια ένα pooling (μέγιστος αριθμός blocks είναι τα 2 - 7x7 εικόνα). Μετά το τελευταίο block ενδύκνειται να γίνονται όσα convolutions-batch normalizations επιθυμεί ο χρήστης χωρίς pooling. Με αυτόν τον τρόπο αυξομειώνουμε τα layers στο μοντέλο όπως εμείς επιθυμούμε. Εκτός από τα layers ρυθμίζονται το batch size, το μέγεθος των φίλτρων και ο αριθμός των φίλτρων στα convolutional layers , το learning rate και ο αριθμός των epochs.
Μετά από πειράματα που κάναμε με διάφορες τιμές για τις υπερπαραμέτρους παρουσιάζουμε τα εξής αποτελέσματα :

- Αριθμός Φιλτρών
  Πραγματοποιήσαμε κάποια πειράματα κρατόντας σταθερές τις υπόλοιπες υπερπαραμέτρους(Μέγεθος φίλτρων 3x3,Learning Rate = 0.001,Batch size = 256) και μεταβάλλοντας τους αριθμούς φίλτρων σε ένα μοντέλο με 3 επίπεδα convolution,παρακάτω βρίσκονται τα αποτελέσματα. Η μικρότερη τιμή του validation loss που είναι και αυτή που θα συγκρίνουμε γενικά αναγράφεται πάνω από το διάγραμμα

Layer (type)	Output Shape	Param #
input_17 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_130 (Conv2D)	(None, 28, 28, 8)	80
batch_normalization_114 (Bat	(None, 28, 28, 8)	32
max_pooling2d_41 (MaxPooling	(None, 14, 14, 8)	0
conv2d_131 (Conv2D)	(None, 14, 14, 16)	1168
batch_normalization_115 (Bat	(None, 14, 14, 16)	64
max_pooling2d_42 (MaxPooling	(None, 7, 7, 16)	0
conv2d_132 (Conv2D)	(None, 7, 7, 32)	4640
batch_normalization_116 (Bat	(None, 7, 7, 32)	128
conv2d_133 (Conv2D)	(None, 7, 7, 32)	9248
batch_normalization_117 (Bat	(None, 7, 7, 32)	128
conv2d_134 (Conv2D)	(None, 7, 7, 16)	4624
batch_normalization_118 (Bat	(None, 7, 7, 16)	64
up_sampling2d_41 (UpSampling	(None, 14, 14, 16)	0
conv2d_135 (Conv2D)	(None, 14, 14, 8)	1160
batch_normalization_119 (Bat	(None, 14, 14, 8)	32
up_sampling2d_42 (UpSampling	(None, 28, 28, 8)	0
conv2d_136 (Conv2D)	(None, 28, 28, 1)	73

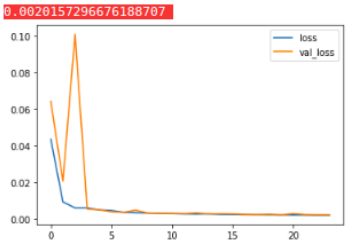
(a) Μοντέλο με αριθμό φιλτρών 8 , 16 , 32



(b) Διάγραμμα loss - validation loss

Layer (type)	Output Shape	Param #
input_20 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_151 (Conv2D)	(None, 28, 28, 16)	160
batch_normalization_132 (Bat	(None, 28, 28, 16)	64
max_pooling2d_47 (MaxPooling	(None, 14, 14, 16)	0
conv2d_152 (Conv2D)	(None, 14, 14, 32)	4640
batch_normalization_133 (Bat	(None, 14, 14, 32)	128
max_pooling2d_48 (MaxPooling	(None, 7, 7, 32)	0
conv2d_153 (Conv2D)	(None, 7, 7, 64)	18496
batch_normalization_134 (Bat	(None, 7, 7, 64)	256
conv2d_154 (Conv2D)	(None, 7, 7, 64)	36928
batch_normalization_135 (Bat	(None, 7, 7, 64)	256
conv2d_155 (Conv2D)	(None, 7, 7, 32)	18464
batch_normalization_136 (Bat	(None, 7, 7, 32)	128
up_sampling2d_47 (UpSampling	(None, 14, 14, 32)	0
conv2d_156 (Conv2D)	(None, 14, 14, 16)	4624
batch_normalization_137 (Bat	(None, 14, 14, 16)	64
up_sampling2d_48 (UpSampling	(None, 28, 28, 16)	0
conv2d_157 (Conv2D)	(None, 28, 28, 1)	145

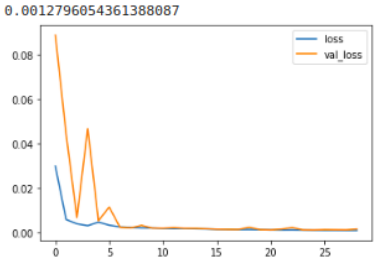
(a) Μοντέλο με αριθμό φιλτρών 16,32,64



(b) Διάγραμμα loss - validation loss

Layer (type)	Output Shape	Param #
input_26 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_193 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_168 (Bat	(None, 28, 28, 32)	128
max_pooling2d_59 (MaxPooling	(None, 14, 14, 32)	0
conv2d_194 (Conv2D)	(None, 14, 14, 64)	18496
batch_normalization_169 (Bat	(None, 14, 14, 64)	256
max_pooling2d_60 (MaxPooling	(None, 7, 7, 64)	0
conv2d_195 (Conv2D)	(None, 7, 7, 128)	73856
batch_normalization_170 (Bat	(None, 7, 7, 128)	512
conv2d_196 (Conv2D)	(None, 7, 7, 128)	147584
batch_normalization_171 (Bat	(None, 7, 7, 128)	512
conv2d_197 (Conv2D)	(None, 7, 7, 64)	73792
batch_normalization_172 (Bat	(None, 7, 7, 64)	256
up_sampling2d_59 (UpSampling	(None, 14, 14, 64)	0
conv2d_198 (Conv2D)	(None, 14, 14, 32)	18464
batch_normalization_173 (Bat	(None, 14, 14, 32)	128
up_sampling2d_60 (UpSampling	(None, 28, 28, 32)	0
conv2d_199 (Conv2D)	(None, 28, 28, 1)	289

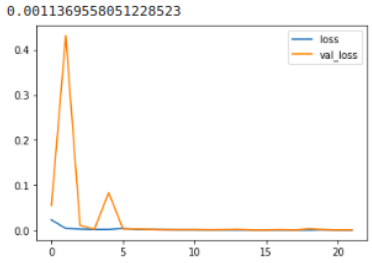
(a) Μοντέλο με αριθμό φιλτρών 32,64,128



(b) Διάγραμμα loss - validation loss

Layer (type)	Output Shape	Param #
input_28 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_207 (Conv2D)	(None, 28, 28, 64)	640
batch_normalization_180 (Bat	(None, 28, 28, 64)	256
max_pooling2d_63 (MaxPooling	(None, 14, 14, 64)	0
conv2d_208 (Conv2D)	(None, 14, 14, 128)	73856
batch_normalization_181 (Bat	(None, 14, 14, 128)	512
max_pooling2d_64 (MaxPooling	(None, 7, 7, 128)	0
conv2d_209 (Conv2D)	(None, 7, 7, 256)	295168
batch_normalization_182 (Bat	(None, 7, 7, 256)	1024
conv2d_210 (Conv2D)	(None, 7, 7, 256)	590080
batch_normalization_183 (Bat	(None, 7, 7, 256)	1024
conv2d_211 (Conv2D)	(None, 7, 7, 128)	295040
batch_normalization_184 (Bat	(None, 7, 7, 128)	512
up_sampling2d_63 (UpSampling	(None, 14, 14, 128)	0
conv2d_212 (Conv2D)	(None, 14, 14, 64)	73792
batch_normalization_185 (Bat	(None, 14, 14, 64)	256
up_sampling2d_64 (UpSampling	(None, 28, 28, 64)	0
conv2d_213 (Conv2D)	(None, 28, 28, 1)	577

(a) Μοντέλο με αριθμό φιλτρών 64,128,256



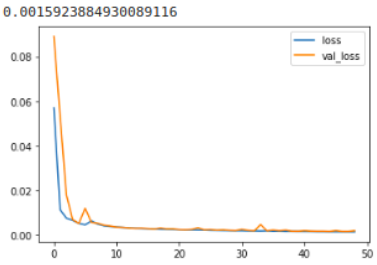
(b) Διάγραμμα loss - validation loss

Συμπεράσματα : Γενικότερα παρατηρούμε πως όσο μεγαλώνει ο αριθμός των φίλτρων τόσο καλύτερο γίνεται το μοντέλο και επιτυγχάνεται μικρότερο validation loss

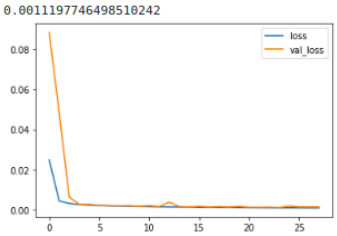
- Μέγεθος Φίλτρου

  Κρατόντας σταθερές τις προηγούμενες παραμέτρους και επιλέγοντας ενδεικτικά ως αριθμούς φιλτρών τα 32,64,128 μεταβάλλαμε το μέγεθος των φίλτρων από 3 x 3 που χρησιμοποιήσαμε στο πρώτο πείραμα σε μικρότερο και μεγαλύτερο και ελέγχουμε τα αποτελέσματα.Η δομή του μοντέλου παραμένει η ίδια

  Τα διαγράμματα που προκύπτουν είναι τα παρακάτω:



(a) Μέγεθος Φίλτρων 2x2

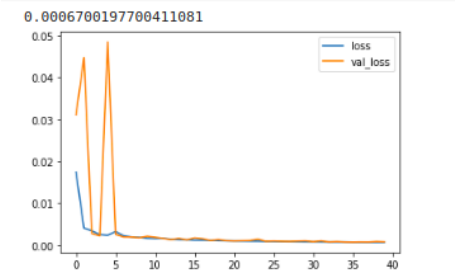


(b) Διάγραμα Φίλτρων 4x4

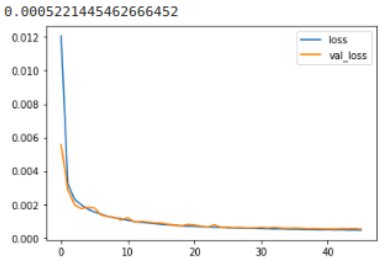
Συμπεράσματα: Συγκρίνοντας τα αποτελέσματα με το 3ο διάγραμμα του 1ου πειράματος για 3x3 filters καταλήγουμε πως όσο αυξάνεται το μέγεθος των φιλτρών τόσο καλύτερο γίνεται το μοντέλο,εξ ου και το μικρότερο validation loss. Δεν μπορούμε να παραβλέψουμε βέβαια πως όσο αυξάνεται το μέγεθος των φιλτρών τόσο αυξάνεται και η υπολογιστική πολυπλοκότητα των βαρών τους, οπότε δεν πρέπει να το παρακάνουμε.

- Batch Size

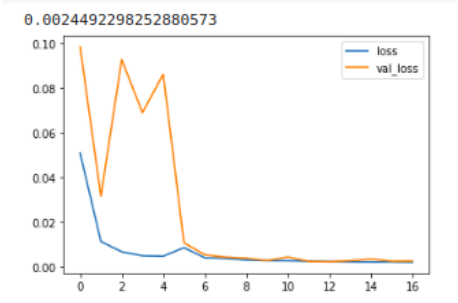
  Συνεχίζουμε τα πειράματα με το batch size . Κρατάμε τις ίδιες παραμέτρους επιλέγοντας μέγεθος φίλτρου 3x3 ενδεικτικά και μεταβάλοντας το batch size από 256 σε 512,128 και 64



(a) Batch Size 128



(b) Batch Size 64

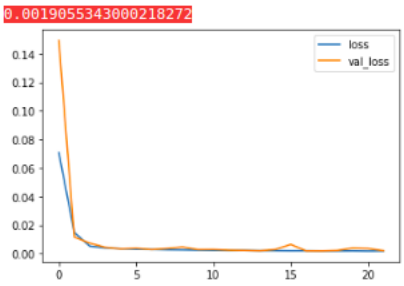


(c) Batch Size 512

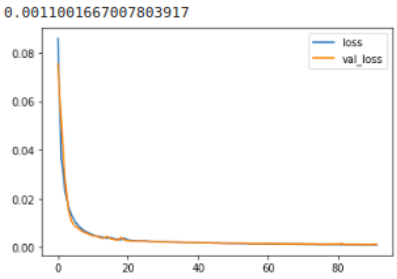
Συμπέρασμα: Παρατηρούμε πως όσο μικρότερο γίνεται το Batch size μας τόσο καλύτερα μαθαίνει το μοντέλο μας . Αυτό είναι λογικό γιατί γίνονται περισσότερα updates .

- Learning Rate

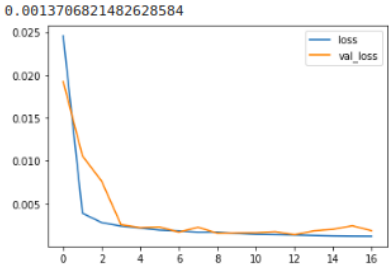
  Συνεχίζουμε με τα πειράματα μεταβάλλοντας το learning rate που μέχρι τώρα ήταν στο 0.001 ενώ ενδεικτικά κρατάμε το batch size στο 256



(a) Learning rate 0.1



(b) Learning rate 0.0001



(c) Learning rate 0.01

Συμπεράσματα: όσο μικρότερο το learning rate τόσο πιο αργά αλλά συγχρόνως και σίγουρα μαθαίνει το μοντέλο μας επειδή γίνονται πιο μικρές μεταβολλές των βαρών. Αυτό παρατηρείται και από τα epochs για τα οποία τρέχει το πρόγραμμα για τις παραμέτρους μας και από το τελικό validation loss.

- Αριθμός Layers

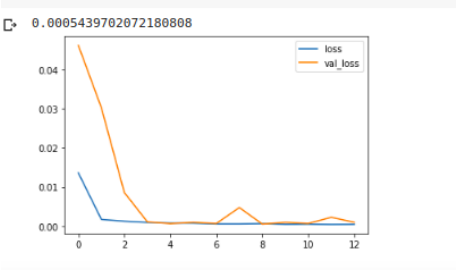
  Εξετάσαμε το μοντέλο σε σχέση με τον αριθμό των layers,τα αποτελέσματα βρίσκονται στην κάτω σελίδα

  Συμπέρασμα: Το validation loss στον auto encoder είναι μικρότερο όταν έχουμε λιγότερα επίπεδα στο μοντέλο μας αλλά αυτό είναι λογικό γιατί δεν κάνουμε αρκετό compression στην αρχική μας εικόνα Δεν θέλουμε να έχουμε ένα μικρό μοντέλο στο B κομμάτι γιατί δεν θα μας οφελήσει γιαυτόν τον λόγο. Με περισσότερα layers το μοντέλο μας γίνεται πιο σύνθετο αλλά δεν υστερεί σε αποδοτικότητα.

Layer (type)	Output Shape	Param #
input_9 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_56 (Conv2D)	(None, 28, 28, 64)	640
batch_normalization_48 (Batc	(None, 28, 28, 64)	256
max_pooling2d_16 (MaxPooling	(None, 14, 14, 64)	0
conv2d_57 (Conv2D)	(None, 14, 14, 128)	73856
batch_normalization_49 (Batc	(None, 14, 14, 128)	512
conv2d_58 (Conv2D)	(None, 14, 14, 128)	147584
batch_normalization_50 (Batc	(None, 14, 14, 128)	512
conv2d_59 (Conv2D)	(None, 14, 14, 64)	73792
batch_normalization_51 (Batc	(None, 14, 14, 64)	256
up_sampling2d_16 (UpSampling	(None, 28, 28, 64)	0
conv2d_60 (Conv2D)	(None, 28, 28, 1)	577

Total params: 297,985
Trainable params: 297,217
Non-trainable params: 768

(a) Λίγα layers

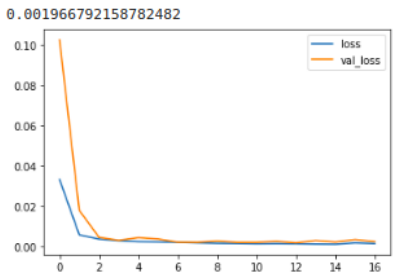


(b) loss-Validation loss

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_72 (Conv2D)	(None, 28, 28, 16)	160
batch_normalization_62 (Batc	(None, 28, 28, 16)	64
conv2d_73 (Conv2D)	(None, 28, 28, 32)	4640
batch_normalization_63 (Batc	(None, 28, 28, 32)	128
max_pooling2d_19 (MaxPooling	(None, 14, 14, 32)	0
conv2d_74 (Conv2D)	(None, 14, 14, 64)	18496
batch_normalization_64 (Batc	(None, 14, 14, 64)	256
conv2d_75 (Conv2D)	(None, 14, 14, 128)	73856
batch_normalization_65 (Batc	(None, 14, 14, 128)	512
max_pooling2d_20 (MaxPooling	(None, 7, 7, 128)	0
conv2d_76 (Conv2D)	(None, 7, 7, 256)	295168
batch_normalization_66 (Batc	(None, 7, 7, 256)	1024
conv2d_77 (Conv2D)	(None, 7, 7, 256)	590080
batch_normalization_67 (Batc	(None, 7, 7, 256)	1024
conv2d_78 (Conv2D)	(None, 7, 7, 128)	295040
batch_normalization_68 (Batc	(None, 7, 7, 128)	512
conv2d_79 (Conv2D)	(None, 7, 7, 64)	73792
batch_normalization_69 (Batc	(None, 7, 7, 64)	256
up_sampling2d_19 (UpSampling	(None, 14, 14, 64)	0
conv2d_80 (Conv2D)	(None, 14, 14, 32)	18464
batch_normalization_70 (Batc	(None, 14, 14, 32)	128
conv2d_81 (Conv2D)	(None, 14, 14, 16)	4624
batch_normalization_71 (Batc	(None, 14, 14, 16)	64
up_sampling2d_20 (UpSampling	(None, 28, 28, 16)	0
conv2d_82 (Conv2D)	(None, 28, 28, 1)	145

Total params: 1,378,433
Trainable params: 1,376,449

(a) Πολλά layers



(b) loss-Validation loss

2 Μερος Β

Με βάση τα παραπάνω συμπεράσματα καταλήγουμε στο ότι για το Β ερώτημα θα χρησιμοποιήσουμε ένα χετικά μεγάλο μοντέλο το οποίο να είναι όσο το δυνατόν πιο αποδοτικό .Θα δοκιμάσουμε 2
Το πρωτο είναι το παρακάτω,επιλέγουμε learning rate = 0.001 και batch size = 128,για τα αποτελέσματα που μας δίνει ο autoencoder. Κόβουμε το μοντέλο που έχει ήδη φορτωμένα τα βάρη στη μέση και το συνδέουμε με ένα dense layer μεγέθους 64,128,256 και με ένα ακόμα μεγέθους 10 και καταλήγουμε στο Classification model. Χρησιμοποιούμε categorial cross-entropy loss loss αφού μετατρέψουμε τα labels σε μορφή one hot.

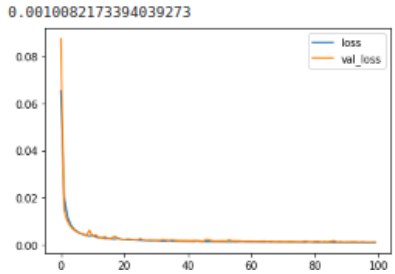
Layer (type)	Output Shape	Param #
input_22 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_299 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_267 (Bat	(None, 28, 28, 32)	128
conv2d_300 (Conv2D)	(None, 28, 28, 32)	9248
batch_normalization_268 (Bat	(None, 28, 28, 32)	128
max_pooling2d_40 (MaxPooling	(None, 14, 14, 32)	0
conv2d_301 (Conv2D)	(None, 14, 14, 64)	18496
batch_normalization_269 (Bat	(None, 14, 14, 64)	256
conv2d_302 (Conv2D)	(None, 14, 14, 64)	36928
batch_normalization_270 (Bat	(None, 14, 14, 64)	256
max_pooling2d_41 (MaxPooling	(None, 7, 7, 64)	0
conv2d_303 (Conv2D)	(None, 7, 7, 128)	73856
batch_normalization_271 (Bat	(None, 7, 7, 128)	512
conv2d_304 (Conv2D)	(None, 7, 7, 128)	147584
batch_normalization_272 (Bat	(None, 7, 7, 128)	512
conv2d_305 (Conv2D)	(None, 7, 7, 256)	295168
batch_normalization_273 (Bat	(None, 7, 7, 256)	1024
conv2d_306 (Conv2D)	(None, 7, 7, 256)	590080
batch_normalization_274 (Bat	(None, 7, 7, 256)	1024
conv2d_307 (Conv2D)	(None, 7, 7, 128)	295040
batch_normalization_275 (Bat	(None, 7, 7, 128)	512
conv2d_308 (Conv2D)	(None, 7, 7, 128)	147584
batch_normalization_276 (Bat	(None, 7, 7, 128)	512
conv2d_309 (Conv2D)	(None, 7, 7, 64)	73792
batch_normalization_277 (Bat	(None, 7, 7, 64)	256
conv2d_310 (Conv2D)	(None, 7, 7, 64)	36928

(a) Autoencoder Model(a)

conv2d_310 (Conv2D)	(None, 7, 7, 64)	36928
batch_normalization_278 (Bat	(None, 7, 7, 64)	256
up_sampling2d_38 (UpSampling	(None, 14, 14, 64)	0
conv2d_311 (Conv2D)	(None, 14, 14, 32)	18464
batch_normalization_279 (Bat	(None, 14, 14, 32)	128
conv2d_312 (Conv2D)	(None, 14, 14, 32)	9248
batch_normalization_280 (Bat	(None, 14, 14, 32)	128
up_sampling2d_39 (UpSampling	(None, 28, 28, 32)	0
conv2d_313 (Conv2D)	(None, 28, 28, 1)	289

Total params: 1,758,657
Trainable params: 1,755,841
Non-trainable params: 2,816

(b) Autoencoder Model(b)



(c) Loss-Validation loss Autoencoder

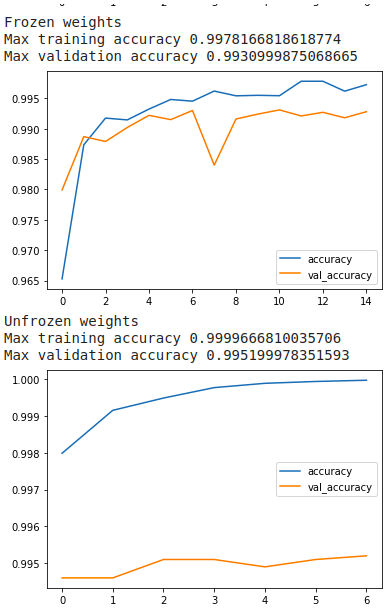
Model: "functional_73"

Layer (type)	Output Shape	Param #
input_22 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_299 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_267 (Bat	(None, 28, 28, 32)	128
conv2d_300 (Conv2D)	(None, 28, 28, 32)	9248
batch_normalization_268 (Bat	(None, 28, 28, 32)	128
max_pooling2d_40 (MaxPooling	(None, 14, 14, 32)	0
conv2d_301 (Conv2D)	(None, 14, 14, 64)	18496
batch_normalization_269 (Bat	(None, 14, 14, 64)	256
conv2d_302 (Conv2D)	(None, 14, 14, 64)	36928
batch_normalization_270 (Bat	(None, 14, 14, 64)	256
max_pooling2d_41 (MaxPooling	(None, 7, 7, 64)	0
conv2d_303 (Conv2D)	(None, 7, 7, 128)	73856
batch_normalization_271 (Bat	(None, 7, 7, 128)	512
conv2d_304 (Conv2D)	(None, 7, 7, 128)	147584
batch_normalization_272 (Bat	(None, 7, 7, 128)	512
conv2d_305 (Conv2D)	(None, 7, 7, 256)	295168
batch_normalization_273 (Bat	(None, 7, 7, 256)	1024
conv2d_306 (Conv2D)	(None, 7, 7, 256)	590080
batch_normalization_274 (Bat	(None, 7, 7, 256)	1024
flatten_15 (Flatten)	(None, 12544)	0
dense_30 (Dense)	(None, 128)	1605760
dense_31 (Dense)	(None, 10)	1290

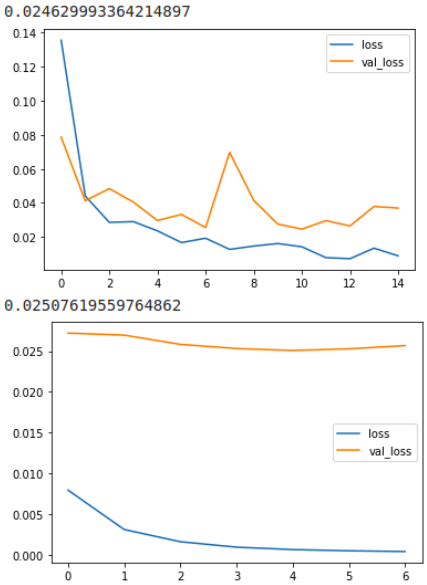
Total params: 2,782,570
Trainable params: 2,780,650
Non-trainable params: 1,920

(d) Classification Model

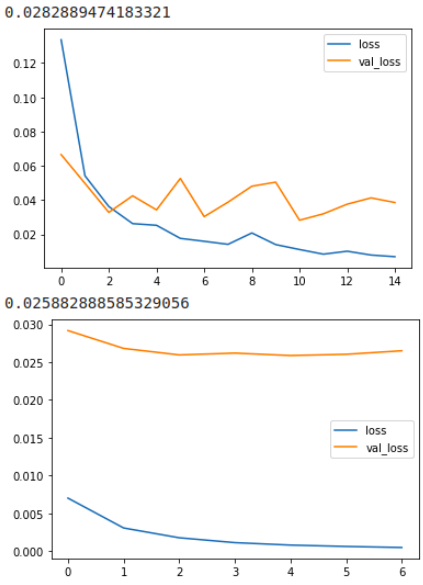
Αρχικά κάνουμε freeze τα βάρη του classification model που πήραμε από τον encoder και στη συνέχεια εκπαιδεύουμε το μοντέλο για τα καινούργια layers(flatten-dense-dense). Βάζουμε ένα μικρό early stopping(4) για να αποφύγουμε κάποιο overfitting και στη συνέχεια κάνουμε unfreeze τα layers και τα εκπαιδεύουμε για λίγα epochs με πολύ μικρότερο learning rate από εκείνο που χρησιμοποιήσαμε πριν έτσι ώστε να μην έχουμε μεγάλη μεταβολή στα ήδη υπάρχοντα βάρη, μέχρι να γίνει επίσης κάποιο early stopping. Τα αποτελέσματα που παίρνουμε βρίσκονται στα διαγράμματα από κάτω.Ως σταθερές υπερπαραμέτρους χρησιμοποιούμε learning rates(0.001 και 0.0001) και batch size 128
Συμπεράσματα για dense = 128: Παρατηρούμε πως με παγωμένα τα βάρη το μοντέλο μας αρχίζει να μαθαίνει(learning rate = 0.001),πέφτει το validation loss και αντίστοιχα να ανεβαίνει το validation accuracy του μοντέλου μας.Όταν το μοντέλο μας σταματάει να μαθαίνει τόσο διακόπτεται η εκπαίδευση και γίνονται unfreeze τα weights με πολύ μικρότερο learning rate(στην περίπτωση μας 0.0001).Στη συνέχεια σταδιακά συνεχίζει να μειώνεται το validation loss και να αυξάνεται το accuracy μέχρι να φτάσει στην μέγιστη τιμή 0.9951 ενώ το μοντέλο μας έχει μάθει επιτυχώς το training set.Οι υπόλοιπες μετρικές(precision,recall) βρίσκονται στα log files.



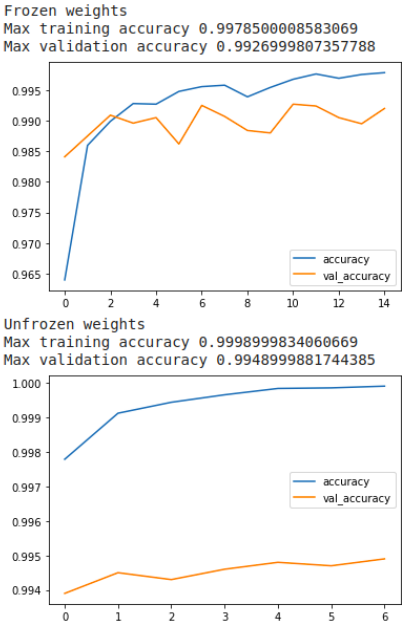
(a) Διαγράμματα accuracy και validation accuracy του classification model για παγωμένα και μη layers



(b) Διάγραμα loss - validation loss για το classification model με παγωμένα και μη layers αντίστοιχα



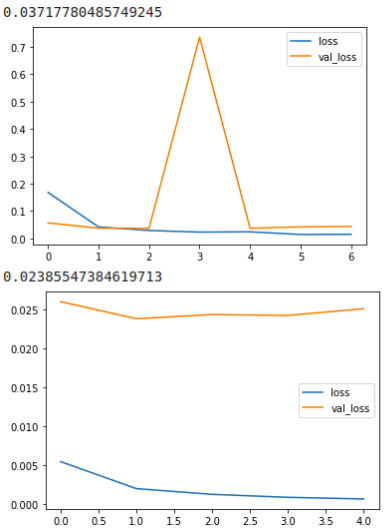
(a) Διαγράμματα accuracy και validation accuracy του classification model για παγωμένα και μη layers



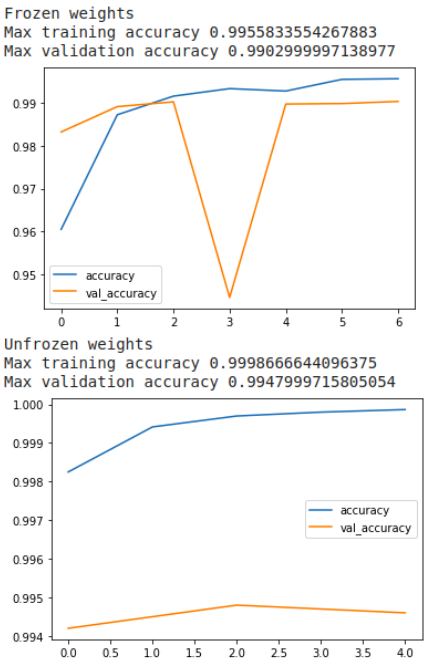
(b) Διάγραμα loss - validation loss για το classification model με παγωμένα και μη layers αντίστοιχα

Επαναλαμβάνουμε το ίδιο πείραμα και αντικαθιστούμε το dense layer σε 64 από 128,παρατηρούμε καλύτερα αποτελέσματα. Στη συνέχεια δοκιμάζουμε και για ένα 2ο πιο μικρό μοντέλο που παρουσιάζουμε πιο κάτω

Επιλέγουμε ως υπερπαραμέτρους για την εκπαίδευση του autoencoder μοντέλου learning rate = 0.001 και batch size = 256 Η διαφορά με το 1ο μοντέλο όσον αφορά το accuracy είναι πολύ μικρή αν και το μοντέλο είναι πολύ μικρότερο. Το πάνω μοντέλο φαίνεται καλύτερο γενικότερα αλλά παρουσιάζεται και αυτό το οποίο έχει ένα επίσης αρκετά καλό accuracy(είναι και σχετικά εύκολο το dataset). Παραδίδουμε το 1ο μοντέλο αλλά μια σημαντική παρατήρηση είναι πως αφού το 2ο μοντέλο κερδίζει σε efficiency και χρόνο training σε κάποιες περιπτώσεις θα μπορούσαμε να προτιμήσουμε και αυτό.



(a) Διάγραμμα loss - validation loss για το classification model με παγωμένα και μη layers αντίστοιχα layers



(b) Διάγραμμα accuracy και validation accuracy του classification model για παγωμενα και μη layers

```
Model: "functional_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_28 (InputLayer)        [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_207 (Conv2D)          (None, 28, 28, 64)        640
_________________________________________________________________
batch_normalization_180 (Bat (None, 28, 28, 64)        256
_________________________________________________________________
max_pooling2d_63 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_208 (Conv2D)          (None, 14, 14, 128)       73856
_________________________________________________________________
batch_normalization_181 (Bat (None, 14, 14, 128)       512
_________________________________________________________________
max_pooling2d_64 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_209 (Conv2D)          (None, 7, 7, 256)         295168
_________________________________________________________________
batch_normalization_182 (Bat (None, 7, 7, 256)         1024
_________________________________________________________________
conv2d_210 (Conv2D)          (None, 7, 7, 256)         590080
_________________________________________________________________
batch_normalization_183 (Bat (None, 7, 7, 256)         1024
_________________________________________________________________
flatten_8 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_16 (Dense)             (None, 128)               1605760
_________________________________________________________________
dense_17 (Dense)             (None, 10)                1290
=================================================================
Total params: 2,569,610
Trainable params: 2,568,202
Non-trainable params: 1,408
_________________________________________________________________
```

(c) Classification model 2 αντίστοιχα

3 Σημειώσεις

- Παραδίδονται 5 αρχεία(2 .py,1 .pynb,1 Readme,2 output file των μοντέλων)
- Συνιστάται η διόρθωση της εργασίας σε Colab
- Τα ορίσματα που χρειάζονται βρίσκονται σε σχόλιο κάτω από το if name == main
- Και στο A και στο B μέρος τα προγράμματα μας κρατάνε early stopping για να αποφευχθεί το overfitting και να σταματάει η εκπαίδευση όταν δεν υπάρχει βελτίωση για κάποιες επαναλήψεις.
- Η εργασία μας έχει υλοποιηθεί και ελεγχθεί πολλές φορές στο Google Colab λόγω GPU(μας έσωσε πολύ χρόνο),χάρην εκφώνησης παραθέτουμε και αρχεία που μεταφέραμε σε python αλλά δεν είναι τόσο τσεκαρισμένη η λειτουργία τους.Παρόλα αυτά θα προσθέσουμε και το .ipynb αρχείο
- Έχουν κρατηθεί tensorboard log files από τα πειράματα που εκτελέσαμε
- Εκτελέσαμε όσο περισσότερα πειράματα μπορούσαμε και μας επέτρεψε η GPU του colab!
- Επειδή η υλοποίηση έγινε στο colab οι υπερπαράμετροι μπαίνουν hard coded στο πρόγραμμα σε μεταβλητές.Παρόλα αυτά, η δημιουργία των autoencoders είναι πλήρως δυναμική