Τελικά Εργασία
Αναγνώρισης Προτύπων

Ονοματεπώνυμο: Κεφαλάς Γεώργιος
ΑΜ: 57406
Μάθημα: Αναγνώριση Προτύπων

Εισαγωγή

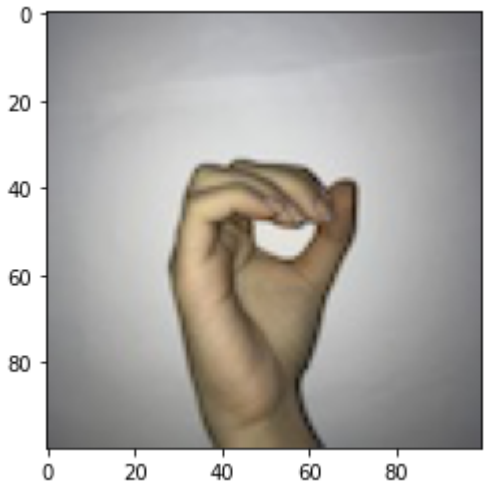
Στην παρούσα εργασία υλοποιείται αλγόριθμος κατά τον οποίο δημιουργείται νευρωνικό δίκτυο για την εκπαίδευση και την ταξινόμηση dataset το οποίο έχει να κάνει την αναπαράσταση αριθμών με τα χέρια για άτομα με δυσκολίες στην ακοή. Πιο συγκεκριμένα αυτό περιέχει εικόνες μεγέθους 100x100x3 για κάθε ένα από τα δεκαδικά νούμερα σε αντίστοιχους φακέλους. Για την επίλυση του προβλήματος χρησιμοποιήθηκε το περιβάλλον του drive για την αποθήκευση των δεδομένων ενώ το "google colab" για την εκτέλεση του αλγορίθμου.

Ανάγνωση και επεξεργασία δεδομένων

Και για τις 2 σκήσεις η ανάγνωση και η επεξεργασία είναι η ίδια καθώς δεν αλλάζει κάτι στον τρόπο και στην μορφα που τα χρειαζόμαστε. Αφού έχουμε εισάγει όλες τις απαραίτητες βιβλιοθήκες μας αρχικοποιούμε 2 κενούς πίνακες Σε αυτούς τους πίνακες αποθηκεύουμε δεδομένα από τις εικόνες του dataset που διαβάζουμε ξεχωριστά από τον κάθε φάκελο που είναι ταξινομημένες και διαχωρισμένες. Έτσι μπορούμε στον δεύτερο πίνακα που δημιουργήσαμε να αποθηκεύσουμε την κλάση του κάθε object ανάλογα σε ποιόν φάκελο είναι αποθηκευμένο. Επειδή, καθόλης την διάρκεια της ανάγνωσης των δεδομένων κάνουμε append τα δεδομένα μας αυτά αποθηκεύονται σε μονοδιάστατη μορφή. Αυτό είναι βολικό μόνο για τον πίνακα με τα label, γι αυτόν το λόγο κάνουμε reshape τον πίνακα x_train ώστε αυτός να έχει όσα object εχει και ο πίνακας y_train αλλά στη μορφή πινάκων με μέγεθος 100x100x3. Έτσι καταλήγουμε να έχουμε δύο πίνακες με μεγέθη 2059x100x100x3 και 2059x1 για τα δεδομένα μας και τις κλάσεις τους αντίστοιχα. Τελικό στάδιο για την επεξεργασία των δεδομένων είναι το να τα κάνουμε normalize διαιρώντας τα με το 255 ώστε αυτά να έχουν τιμές από 0 έως -1.

Στη συνέχεια θέλοντας να δημιουργήσουμε ένα testing set, παίρνουμε "τυχαία" αντικείμενα από το training set μας και τα αποθηκεύουμε σε μια άλλη μεταβλητή x_test με τη χρήση μιας επανάληψης for η οποία προσπελαύνει τα objects του x_train. Επειδή και σε αυτή την περίπτωση χρησιμοποιούμε την εντολή append ακολουθούμε την ίδια διαδικασία με πριν ώστε να έχουμε τα δεδομένα στη μορφή που τα θέλουμε, ενώ τέλος τα κάνουμε και αυτά normalize. Το testing set θα μας χρειαστεί για νε ελέγξουμε την απόδοση του αλγορίθμου μας αφού ολοκληρωθεί η εκπαίδευση.

Τέλος με την συνάρτηση imshow εμφανίζουμε μια εικόνα ώστε να επαληθεύσουμε ότι το dataset μας έχει επεξεργαστεί σωστά.



Άσκηση 1η

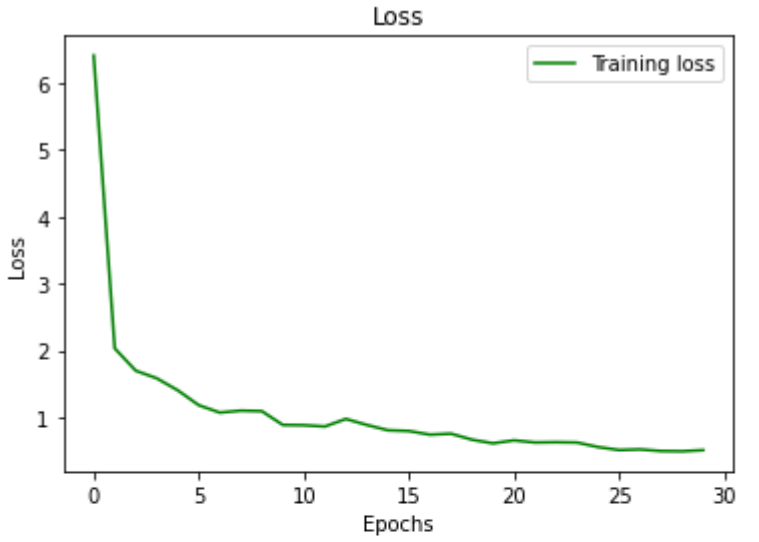
- Ερώτημα α)

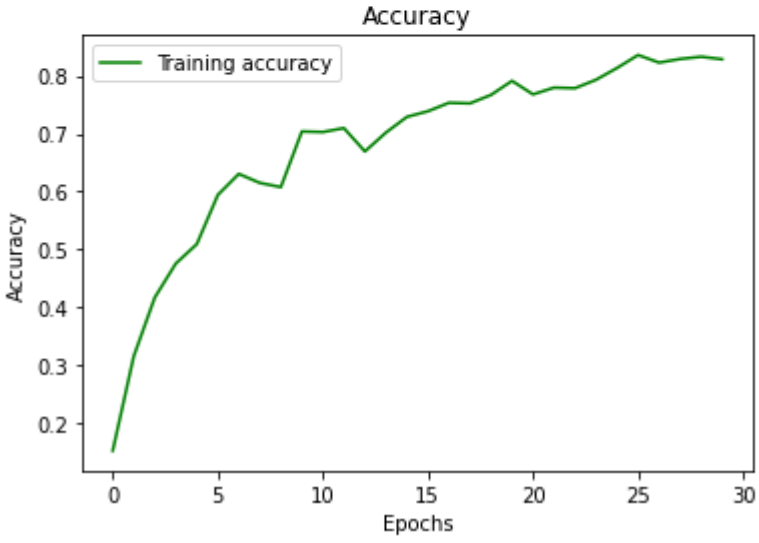
Για το πρώτο ερώτημα, αρχικά δημιουργούμε ένα νευρωνικό δίκτυο με βάση τα δεδομένα της εκφώνησης καθώς επίσης και με τη μορφή των δεδομένων που έχουμε. Οπότε, καταλήγουμε στο να έχουμε ένα layer εισόδου της με 30000 εισόδους καθώς έχουμε εικόνες 100x100x3 = 3000 ενώ στη συνέχεια έχουμε 2 hidden layers με 256 και 128 νευρώνες το κάθε ένα αντίστοιχα. Τέλος καταλήγουμε σε ένα layer εξόδου με 10 νευρώνες καθώς έχουμε 10 κλάσεις βάση των οποίων γίνεται η ταξινόμηση. Παρακάτω φαίνεται και το summary του δικτύου.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 30000)             0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               7680256
_________________________________________________________________
dense_7 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
=================================================================
Total params: 7,714,442
Trainable params: 7,714,442
Non-trainable params: 0
_________________________________________________________________
```

Στη συνέχεια κάνουμε compile το δίκτυο που έχουμε δημιουργήσει χρησιμοποιώντας τον adam optimizer και έχοντας ώς μετρική την ακρίβεια. Αφού ολοκληρωθεί αυτό, τότε ξεκινάμε το fit του μοντέλου μας ώστε να πραγματοποιηθεί η εκπαίδευση. Η διαδικασία αυτή παρατηρούμε να σταματάει μετά από **30 εποχές** με βάση το earlystopping το οποίο έχουμε ορίσει εμείς οι ίδιοι. Η ακρίβεια που φαίνεται να πετυχαίνουμε είναι εν τέλη **82.6%**. Για το validation set, δεν μπορέσαμε να βγάλουμε κάποια ακρίβεια καθώς αυτό απαιτεί να είναι τα δεδομένα μας μπερδεμένα. Αυτό γίνεται καθώς κατά την εκπαίδευση δίνουμε ένα ποσοστό σαν validation set από το training set το οποίο και παίρνει ο αλγόριθμος από τα τελευταία δεδομένα του training set. "Ανακατεύοντας" λοιπόν τα δεδομένα μας, θα μπορούσαμε να πετύχουμε καλύτερα αποτελέσματα και στο accuracy/loss για το validation. Παρακάτω φαίνονται και τα αποτελέσματα με τα γραφήματα που ζητούνται από την εκφώνηση της άσκησης.

```
Epoch 22/30
65/65 [==============================] - 0s 5ms/step - loss: 0.6689 - accuracy: 0.7655
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 23/30
65/65 [==============================] - 0s 5ms/step - loss: 0.5964 - accuracy: 0.7894
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 24/30
65/65 [==============================] - 0s 4ms/step - loss: 0.5954 - accuracy: 0.8034
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 25/30
65/65 [==============================] - 0s 5ms/step - loss: 0.5439 - accuracy: 0.8118
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 26/30
65/65 [==============================] - 0s 4ms/step - loss: 0.5656 - accuracy: 0.8236
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 27/30
65/65 [==============================] - 0s 5ms/step - loss: 0.5490 - accuracy: 0.8139
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 28/30
65/65 [==============================] - 0s 5ms/step - loss: 0.4911 - accuracy: 0.8300
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 29/30
65/65 [==============================] - 0s 4ms/step - loss: 0.4867 - accuracy: 0.8305
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 30/30
65/65 [==============================] - 0s 4ms/step - loss: 0.5107 - accuracy: 0.8263
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
```





Τέλος, εφαρμόζοντας την εκπαίδευση που έχουμε ήδη κάνει παρατηρούμε το accuracy που πετυχαίνουμε όταν ταξινομούμε το training set. Βλέπουμε ότι αυτό έχει ακρίβεια της τάξης του **86.4%**.

- Ερώτημα b)

Κατά την εκπαίδευση πολλές φορές είναι πολύ πιθανό να εντοπιστει το φαινόμενο του overfitting. Σε αυτή την περίπτωση η εκπαιδευση είναι πιο εξειδικευμένη στα χαρακτηριστικά των δεδομένων μας και δεν είναι γενικευμενα για περισσότερες περιπτώσεις πράγμα που μπορεί να οδηγήσει σε λανθασμένα αποτελέσματα στο μέλλον. Για την αντιμετώπιση αυτού του προβλήματος εμείς θα μπορούσαμε να κάνουμε κάποια βήματα ώστε να αποφύγουμε τέτοιου είδους καταστάσεις. Αυτά τα βήματα θα μπορούσαν να είναι για παράδειγμα η χρήση dropout στα layers του

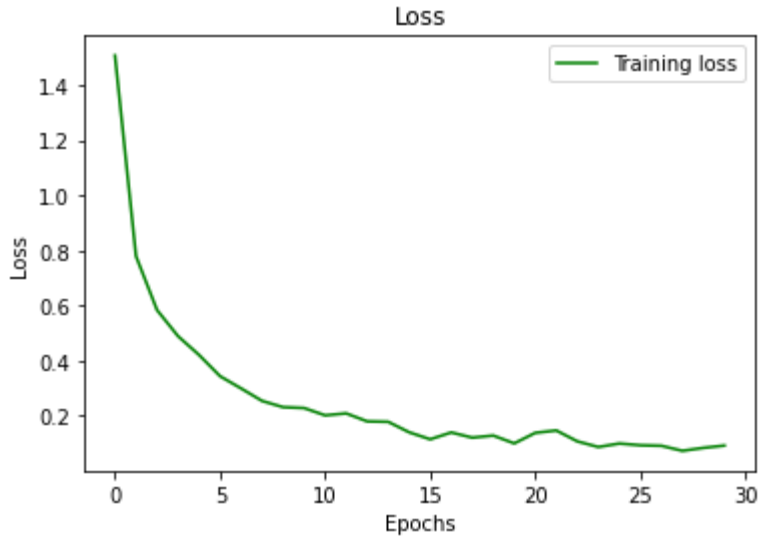
νευρωνικού για να αποφυγουμε χαρακτηριστικά που παραπλανούν την εκπαίδευση. Ακόμα πολλές φορές βοηθάει και η χρήση απλούστερων δικτύων που δεν μπερδεύουν την εκπαίδευση με πολλά πολλές φορές αχρείαστα layer. Τέλος πολύ χρήσιμο μπορεί να φανεί και το earlystopping που αποτρέπει να φτάσουμε σε αυτό το σημείο καθώς σταματάει την διαδικασία της εκπαίδευσης πριν εμφανιστεί κάποιου είδους overfit. Στην περίπτωση μας, παρατηρώντας αρκετά μεγάλο αριθμό διακυμάνσεων στις γραφικές παραστάσεις, μπορούμε να συμπεράνουμε ότι το overfitting έχει παίξει σημαντικό ρόλο στην εκπαίδευση του dataset μας.

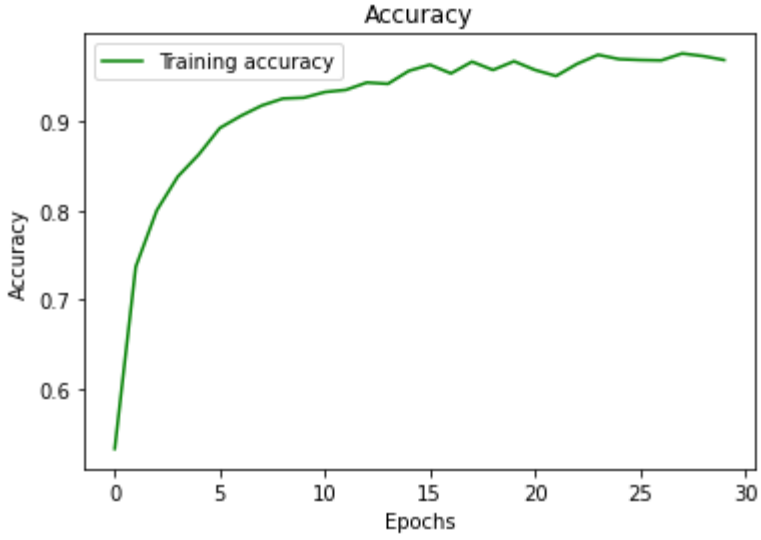
- Ερώτημα c)

Στο 3ο υποερώτημα της πρώτης ασκησης εξετάζουμε την εφαρμογή ενός activation function και συγκεκριμένα του ReLu στα hidden layers του νευρωνικού μας δικτύου. Έχοντας ήδη έτοιμα τα δεδομένα από τα προηγούμενα κιόλας ερωτήματα, αρχικοποιούμε ένα καινούργιο δίκτυο με τον ίδιο αριθμό νευρώνων σε κάθε layer όπως και στο προηγούμενο. Η διαφορά είναι ότι σε τώρα έχουμε την ReLu. Αφού λοιπόν ορίσουμε το δίκτυο μας και το κάνουμε compile, περνάμε στην εκπαίδευση την οποία ολοκληρώνουμε όπως ακριβως και προηγουμένως. Στο τέλος παρατηρούμε έναν πολύ μικρότερο αριθμό διακυμάνσεων πράγμα, που μας οδηγεί στο συμπέρασμα ότι το μοντέλο μας είναι πιο γενικευμένο σε σχέση με πριν και όχι τόσο ορισμένο πάνω στα δεδομένα που έχουμε. Αυτό μας οδηγεί και στο συμπέρασμα ότι με τη χρήση της συνάρτησης ReLu το δίκτυο μας δεν είναι τόσο επιρρεπές στο overfitting. Έπειτα από την εκπαίδευση επίσης παρατηρούμε ότι η ακρίβεια είναι στα ίδια επίπεδα σε σχέση με πριν στο **96.8%** ενώ έχουμε επίσης **30 εποχές** που χρειάστηκαν για την εκπαίδευση.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_5 (Flatten)          (None, 30000)             0
_________________________________________________________________
dense_15 (Dense)             (None, 256)               7680256
_________________________________________________________________
dense_16 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_17 (Dense)             (None, 10)                1290
=================================================================
Total params: 7,714,442
Trainable params: 7,714,442
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 23/30
65/65 [==============================] - 0s 6ms/step - loss: 0.1076 - accuracy: 0.9641
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 24/30
65/65 [==============================] - 0s 7ms/step - loss: 0.0871 - accuracy: 0.9743
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 25/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0998 - accuracy: 0.9694
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 26/30
65/65 [==============================] - 0s 7ms/step - loss: 0.0935 - accuracy: 0.9684
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 27/30
65/65 [==============================] - 0s 7ms/step - loss: 0.0915 - accuracy: 0.9679
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 28/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0733 - accuracy: 0.9757
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 29/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0840 - accuracy: 0.9728
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 30/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0924 - accuracy: 0.9684
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
```





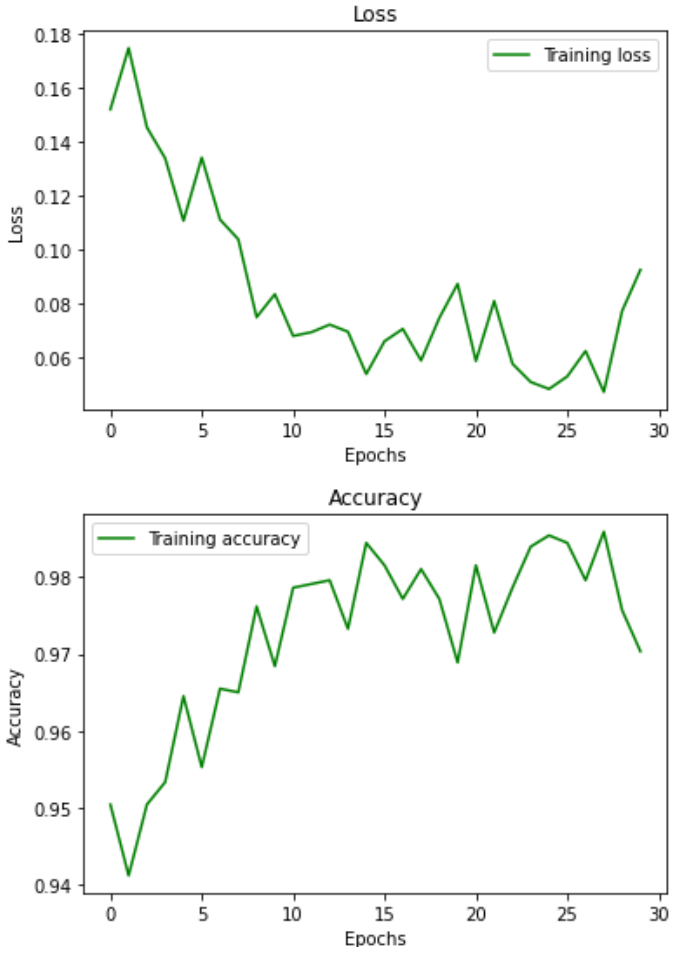
- Ερώτημα d)

Όπως και στο προηγούμενο ερώτημα διατηρούμε ίδιο τον κορμό του νευρωνικού μας δικτύου και αυτή τη φορά προσθέτουμε **batch normalization** σε κάθε επίπεδο. Επιπλέον, για έξτρα βελτιστοποίηση όσον αφορά το overfitting προσθέτουμε και **dropout** μετά από κάθε layer. Με την προσθήκη αυτών το δίκτυο μας βελτιστοποιείται από άποψη ταχύτητας, επίδοσης και σταθερότητας. Με τη βοήθεια αυτού κάθε layer κρατάει τις μέσες τιμές των μεταβολών των βαρών του πράγματα που βοηθάει στο να μην υπάρχουν μεγάλες διαταράξεις στα βάρη με συνέπεια να γίνεται το δίκτυο μας πιο σταθερό. Έτσι τα layers του νευρωνικου μας μαθαίνουν στα βαθύτερα layers και το overfitting εμφανίζεται λιγότερο σε σχέση με πριν. Επίσης, με την εφαρμογή του dropout σε κάθε layer αυτό που πετυχαίνουμε είναι να μειώνουμε ακόμα περισσότερο το φαίνομενο του overfitting. Πιο συγκεκριμένα, με αυτο, το δίκτυο μας γίνεται πιο επιλεκτικό στο να επιλέγει συγκεκριμένα μονοπάτια στο δίκτυο ενώ ταυτόχρονα απορρίπτει άλλα λιγότερο σημαντικά. Η πιθανότητα αυτή ορίζεται από εμάς πράγμα που κάναμε εμείς δίνοντας του την τιμή 0.2. Αυτή η απόρριψη οδηγεί στο να μην έχουμε συνδέσεις με πολύ μεγάλα βάρη και έτσι καταλήγουμε σε μικρότερο δίκτυο το οποίο μοιράζεται τα βάρη στους νευρώνες ισορροπημένα. Παρακάτω ορίζεται η νέα αρχικοποίηση που κάναμε στο δίκτυο μας βάση αυτών που προστέθηκαν σε αυτό το ερώτημα. Στη συνέχεια ακολουθείται η ίδια διαδικασία κατά την οποία κάνουμε compile και fit. Έπειτα στη συνέχεια απεικονίζουμε και τα αποτελέσματα της εκπαίδευσης με αυτες τις προσθήκες. Παρακάτω φαίνονται αναλυτικά τα αποτελέσματα.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_10 (Flatten)         (None, 30000)             0
_________________________________________________________________
batch_normalization_12 (Batc (None, 30000)             120000
_________________________________________________________________
dense_30 (Dense)             (None, 256)               7680256
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_13 (Batc (None, 256)               1024
_________________________________________________________________
dense_31 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_9 (Dropout)          (None, 128)               0
_________________________________________________________________
batch_normalization_14 (Batc (None, 128)               512
_________________________________________________________________
dense_32 (Dense)             (None, 10)                1290
=================================================================
Total params: 7,835,978
Trainable params: 7,775,210
Non-trainable params: 60,768
_________________________________________________________________
```

Σε αυτό το σημείο, έπειτα από την εκπαίδευση παρατηρούμε απόδοση της τάξης του **97.5%** ενώ ταυτόχρονα έχουμε μεγάλη σταθερότητα στα αποτελέσματα της εκπαίδευσης.

```
Epoch 23/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0679 - accuracy: 0.9755
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 24/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0581 - accuracy: 0.9837
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 25/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0387 - accuracy: 0.9871
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 26/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0608 - accuracy: 0.9794
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 27/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0554 - accuracy: 0.9806
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 28/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0371 - accuracy: 0.9891
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 29/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0695 - accuracy: 0.9762
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 30/30
65/65 [==============================] - 0s 6ms/step - loss: 0.0915 - accuracy: 0.9705
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
```



- Ερώτημα e)

Σε αυτό το ερώτημα δοκιμάσαμε διάφορες προσθήκες και παραλλαγές στο νευρωνικό μας δίκτυο ώστε να προσπαθήσουμε να πετύχουμε όσο το δυνατόν περισσότερη ακρίβεια στην εκπαίδευση και ταξινόμηση του training set και το testing set αντίστοιχα. Στον παρακάτω πίνακα φαίνονται οι διάφορες αρχιτεκτονικές που υλοποιήθηκαν μαζί με τα αποτελέσματα της κάθε μιας ώστε να βρούμε αυτή η οποία μας παρέχει με τα βέλτιστα αποτελέσματα.

Με τη χρήση της **ReLu**

Αρχιτεκτονικη	Layer 1	Layer 2	Layer 3	Batch Size.	Dropout	Accuracy
1	256	256	-	32	0.2	97.11%
2	256	256	-	64	0.4	93.55%
3	512	512	-	32	0.2	98.37%
4	512	512	-	64	0.4	95.73%
5	256	256	256	32	0.2	96.96%
6	512	512	512	32	0.2	96.93%
7	256	512	256	32	0.2	95.84%

Με τη χρήση της **Selu**

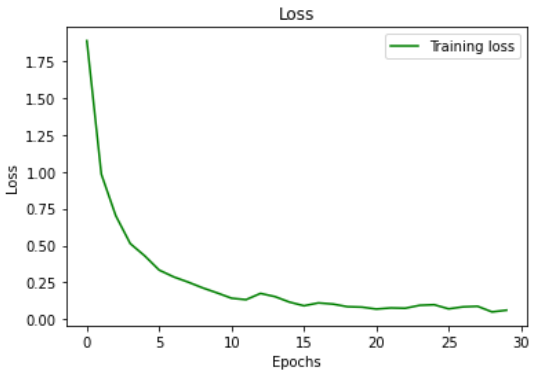
Αρχιτεκτονικη	Layer 1	Layer 2	Layer 3	Batch Size.	Dropout	Accuracy
1	256	256	-	32	0.2	97.49%
2	256	256	-	64	0.4	96.32%
3	512	512	-	32	0.2	97.88%
4	512	512	-	64	0.4	97.41%
5	256	256	256	32	0.2	97.05%
6	512	512	512	32	0.2	97.09%

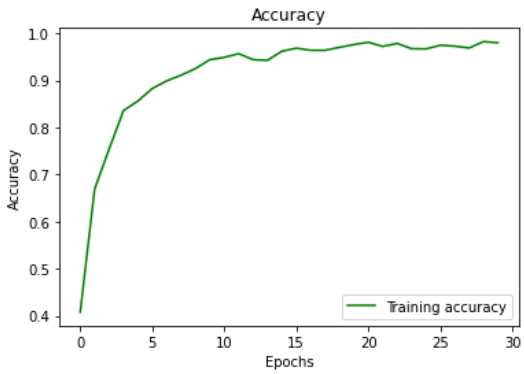
Με τη χρήση της **ELU**

Αρχιτεκτονικη	Layer 1	Layer 2	Layer 3	Batch Size.	Dropout	Accuracy
1	128	128	128	32	0.2	95.73%
2	256	256	-	32	0.2	96.62%
3	256	256	-	64	0.4	94.75%
4	512	512	-	32	0.2	97.88%
5	512	512	-	64	0.4	97.91%
6	256	256	256	32	0.2	96.03%
7	512	512	512	32	0.2	96.80%

➔ Αρχιτεκτονική **ReLu_3**

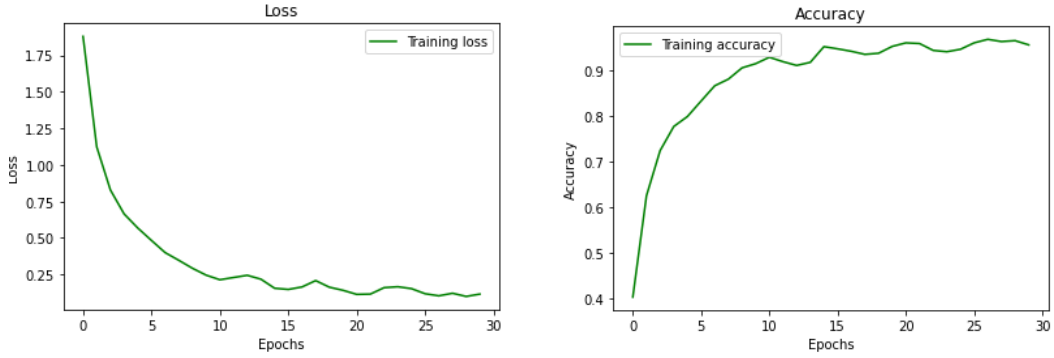
```
Epoch 24/30
65/65 [==============================] - 1s 8ms/step - loss: 0.0794 - accuracy: 0.9724
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 25/30
65/65 [==============================] - 1s 8ms/step - loss: 0.1079 - accuracy: 0.9666
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 26/30
65/65 [==============================] - 1s 8ms/step - loss: 0.0718 - accuracy: 0.9755
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 27/30
65/65 [==============================] - 0s 8ms/step - loss: 0.0862 - accuracy: 0.9747
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 28/30
65/65 [==============================] - 1s 8ms/step - loss: 0.0826 - accuracy: 0.9657
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 29/30
65/65 [==============================] - 1s 8ms/step - loss: 0.0467 - accuracy: 0.9823
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 30/30
65/65 [==============================] - 0s 8ms/step - loss: 0.0512 - accuracy: 0.9837
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
```





➔ Αρχιτεκτονική **ELU_5**

```
Epoch 24/30
33/33 [==============================] - 0s 10ms/step - loss: 0.0633 - accuracy: 0.9811
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 25/30
33/33 [==============================] - 0s 10ms/step - loss: 0.0578 - accuracy: 0.9820
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 26/30
33/33 [==============================] - 0s 10ms/step - loss: 0.0557 - accuracy: 0.9801
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 27/30
33/33 [==============================] - 0s 10ms/step - loss: 0.0452 - accuracy: 0.9845
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 28/30
33/33 [==============================] - 0s 10ms/step - loss: 0.0424 - accuracy: 0.9854
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 29/30
33/33 [==============================] - 0s 10ms/step - loss: 0.0845 - accuracy: 0.9767
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
Epoch 30/30
33/33 [==============================] - 0s 10ms/step - loss: 0.0576 - accuracy: 0.9791
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
```



Από τις διάφορες αρχιτεκτονικές χρησιμοποιώντας διαφορετικό αριθμό επιπέδων, με διαφορετικό αριθμό νευρώνων. διαφορετικά χαρακτηριστικά αλλά και άλλες συναρτήσεις ενεργοποίησης, καταφερα να να φτάσουμε την μέγιστη απόδοση με ποσοστό ακρίβειας **98.37%.**

Για την καλύτερη αρχιτεκτονική χρησιμοποιήσαμε 2 layer με 512 νευρώνες το κάθε ένα. Επιπλέον, αυτά είχαν την συνάρτηση ενεργοποίησης ReLu που παίζει σημαντικό ρόλο στη χρήση του back propagation αλγόριθμου καθώς είναι γραμμική και μηδενίζει τις τιμές μικρότερες ή ίσες του μηδέν. Επίσης η ReLu βοηθάει και στην αποφυγή του vanishing gradient. Ακόμα, έγινε η χρήση του softmax function για το εξωτερικο layer. Άλλο ένα χαρακτηριστικό που έπαιξε ρόλο στην καλύτερη αρχιτεκτονική ήταν και το dropout με πιθανοτητα 0.2 για απόρριψη νευρώνων που ήταν μικρότερη σε σχέση με άλλες εφαρμογές του νευρωνικού δικτύου. Τέλος για όλα χρησιμοποιηθηκε ο adam

optimizer πράγμα που δεν επηρεάζει συγκριτικά την απόδοση σε σχέση με άλλες εκτελέσεις αλλά προσδίδει στο δίκτυο μας αποτελεσματικότητα, χαμηλή απαίτηση στην μνήμη καθώς επίσης και ο ρυθμός εκπαίδευσης του δεν μένει αμετάβλητος αλλά για κάθε παράμετρο έχει διαφορετικό πράγμα που βοηθάει στην απόδοση της εκπαίδευσης.

Άσκηση 2η

- Ερώτημα α)

Για το πρώτο ερώτημα της 2ης άσκησης, αρχικά εισάγουμε τα δεδομένα μας αφού πρώτα έχουμε δηλώσει τις βιβλιοθήκες τις οποίες θα χρησιμοποιήσουμε. Τα δεδομένα εισάγονται με τον ίδιο ακριβώς τρόπο όπως έγινε και στην προηγούμενη άσκηση καθώς χρησιμοποιούμε ακριβώς το ίδιο dataset. Αφού ολοκληρωθεί και η εισαγωγή των δεδομένων, προχωράμε στην επεξεργασία αυτών η οποια γίνεται επίσης με τον ίδιο τρόπο σε σχέση με πριν ώστε τελικά να έχουμε τις εικόνες του dataset στη μορφή που τις θέλουμε εν τέλη. Αυτό ολοκληρώνεται με το να έχουμε τελικά έναν πίνακα εκπαίδευσης με 2059 αντικείμενα με την μορφή εικόνων 3d με μέγεθος 100x100x3. Αφού ολοκληρωθεί και αυτό το κομμάτι τότε, παίρνουμε δεδομένα από τον πίνακα του train ώστε να δημιουργήσουμε και μια μεταβλητή με δεδομένα για testing. Έπειτα κάνουμε normalize τα δεδομένα μας με αποτέλεσμα να τελειώσουμε το κομμάτι της προετοιμασίας τους για να την εκπαίδευση. Τέλος εκτυπώνουμε την πρώτη εικόνα του dataset για να βεβαιωθούμε ότι όλα έχουν γίνει όπως τα θέλαμε.

Έπειτα συνεχίζουμε με την δημιουργία του συνελικτικού νευρωνικού δικτύου το οποίο ορίζεται με βάση τα δεδομένα της εκφώνης. Επομένως καταλήγουμε έχουμε 2 επίπεδα συνέλιξης με 64 και 32 νευρώνες αντίστοιχα και με μέγεθος παραθύρου 3x3. :Επιπλέον χρησιμοποιούμε και ένα layer εξόδου ώστε να έχουμε τον αριθμό των εξόδων που θέλουμε. Αφού ολοκληρωθεί και το στάδιο της δημιουργίας του δικτύου έπειρα προχωράμε στο να το κάνουμε compile και έπειτα να ξεκινήσουμε την εκπαίδευση με με τα δεδομένα τα οποία έχουμε.

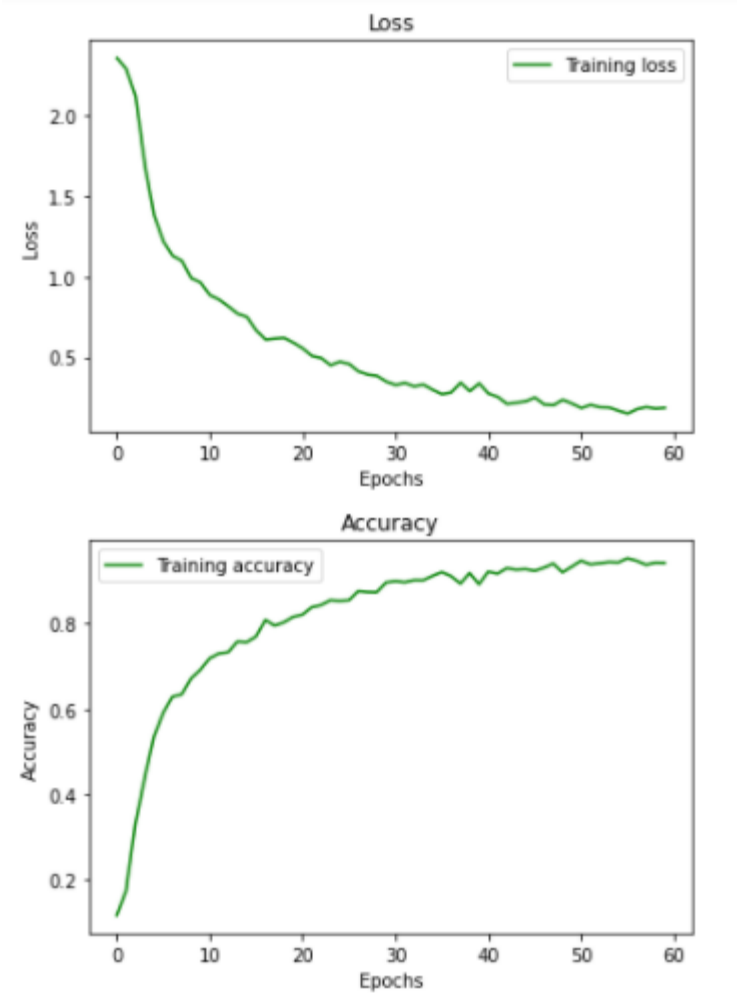
Παρακάτω φαίνεται το summary του δικτύου που δημιουργήθηκε εν τελη για την εκπαίδευση του δικτύου.

```
model = Sequential()

#First Hidden Layer
model.add(Conv2D(64,(3,3),input_shape=(100,100,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

#Second Hidden Layer
model.add(Conv2D(32,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.2))

model.add(Dense(10))
model.add(Activation('softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 47, 47, 32)        18464
_________________________________________________________________
activation (Activation)      (None, 47, 47, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 16928)             0
_________________________________________________________________
dense (Dense)                (None, 128)               2166912
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 2,188,458
Trainable params: 2,188,458
Non-trainable params: 0
_________________________________________________________________
```

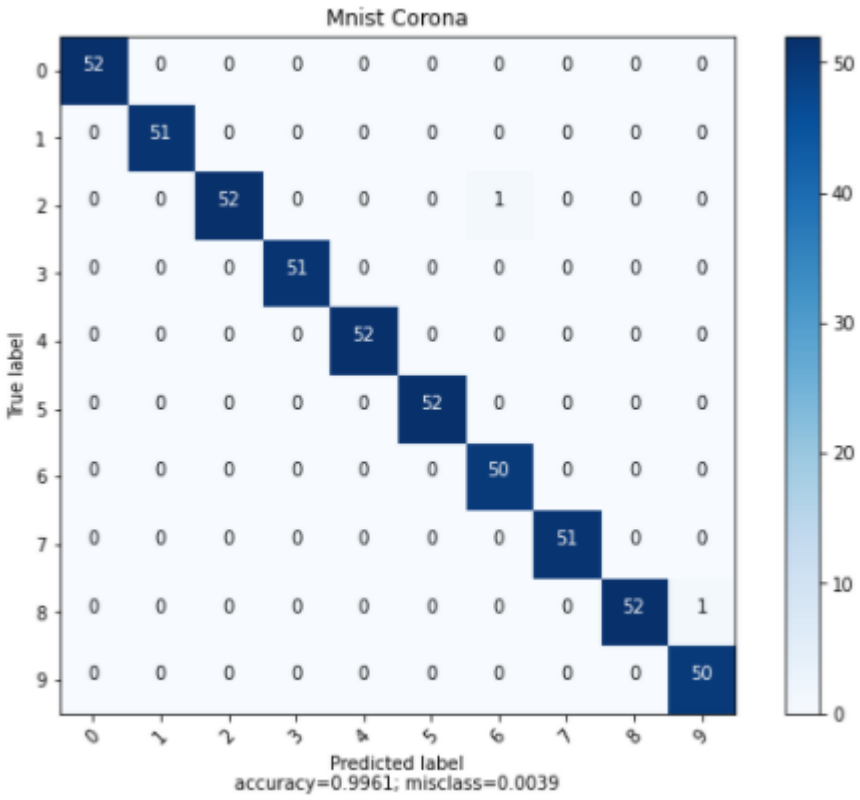
- Ερώτημα b)

Αφού ολοκληρώθηκε η εκπαίδευση παρατηρήθηκε ότι υπήρχε ακρίβεια με μέγεθος 94%. Πιο συγκεκριμένα, παρατηρήθηκε ότι μετά τις 45 περίπου εποχές υπάρχει μια σταθεροποίηση της τιμής της ακρίβειας μεταξύ 94-96% πράγμα που δηλώνει ότι και αυτός είναι και ο βέλτιστος αριθμός εποχών. Με βάση αυτόν, υπολογίστηκαν και τα παρακάτω αποτελέσματα.




```
score = model.evaluate(x_test,  y_test)

print('\nScore', score)
```

```
17/17 [==============================] - 0s 5ms/step - loss: 0.0277 - accuracy: 0.9961

Score [0.027738576754927635, 0.9961165189743042]
```



Επιπλέον, δεν παρατηρείται ιδιαίτερα το φαινόμενο του overfitting και αυτό επειδή δεν υπάρχουν αρκετές διακυμάνσεις κατά την εκπαίδευση πράγμα που θα σήμαινε ότι το δίκτυο μας εκπαιδεύεται μόνο με βάση τα αντικείμενα τα οποία υπάρχουν στο training set και δεν υπάρχει κάποια γενικευμένη λύση που να καλύπτει οποιοδήποτε object δοθεί για να ταξινομηθεί.

- Ερώτημα c)

Παρακάτω φαίνονται διάφορες δομές συνελικτικών νευρωνικών δικτύων ο οποίες δοκιμάστηκαν μαζί με τα αποτελέσματα τους.

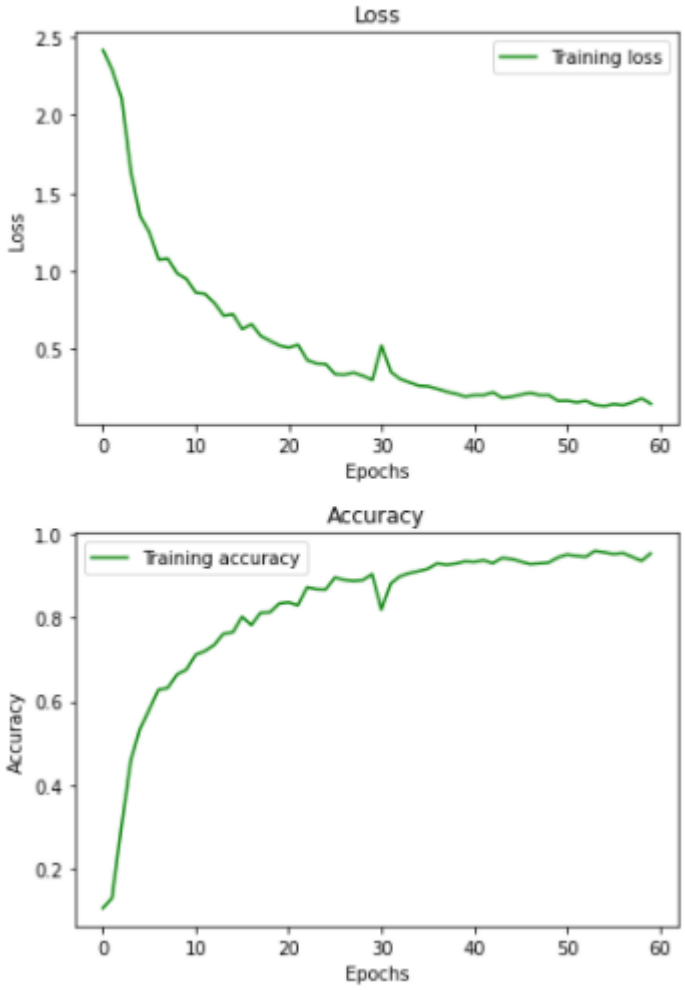
➔ Επίπεδα συνέλιξης με 128 και 64 φίλτρα

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 98, 98, 128)       3584
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 49, 49, 128)       0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 47, 47, 64)        73792
_________________________________________________________________
activation_12 (Activation)   (None, 47, 47, 64)        0
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               4333696
_________________________________________________________________
activation_13 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_14 (Activation)   (None, 10)                0
=================================================================
Total params: 4,412,362
Trainable params: 4,412,362
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 57/60
16/16 [==============================] - 4s 256ms/step - loss: 0.1749 - accuracy: 0.9399
Epoch 58/60
16/16 [==============================] - 4s 257ms/step - loss: 0.1820 - accuracy: 0.9449
Epoch 59/60
16/16 [==============================] - 4s 254ms/step - loss: 0.1858 - accuracy: 0.9325
Epoch 60/60
16/16 [==============================] - 4s 256ms/step - loss: 0.1570 - accuracy: 0.9497
```

```
[56] score = model.evaluate(x_test,  y_test)

     print('\nScore', score)

     17/17 [==============================] - 0s 7ms/step - loss: 0.0205 - accuracy: 0.9981

     Score [0.020539145916700363, 0.9980582594871521]
```

Τα διαγράμματα των αποτελεσμάτων:



Σε αυτή την εκτέλεση παρατηρείτε ελάχιστα το φαινόμενο του overfitting.

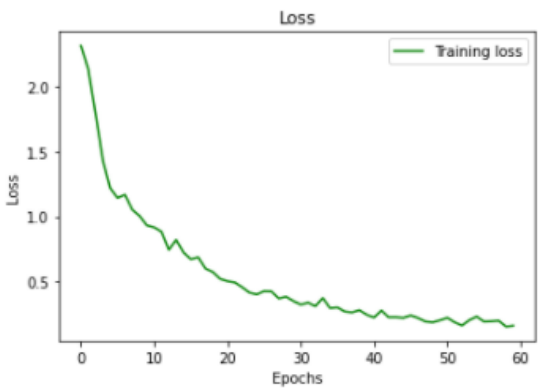
➔ Batch size = 64

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_14 (Conv2D)           (None, 98, 98, 64)        1792
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 47, 47, 32)        18464
_________________________________________________________________
activation_21 (Activation)   (None, 47, 47, 32)        0
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 23, 23, 32)        0
_________________________________________________________________
flatten_7 (Flatten)          (None, 16928)             0
_________________________________________________________________
dense_14 (Dense)             (None, 128)               2166912
_________________________________________________________________
activation_22 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_15 (Dense)             (None, 10)                1290
_________________________________________________________________
activation_23 (Activation)   (None, 10)                0
=================================================================
Total params: 2,188,458
Trainable params: 2,188,458
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 54/60
16/16 [==============================] - 2s 135ms/step - loss: 0.2344 - accuracy: 0.9294
Epoch 55/60
16/16 [==============================] - 2s 136ms/step - loss: 0.2351 - accuracy: 0.9275
Epoch 56/60
16/16 [==============================] - 2s 137ms/step - loss: 0.1945 - accuracy: 0.9353
Epoch 57/60
16/16 [==============================] - 2s 130ms/step - loss: 0.1902 - accuracy: 0.9394
Epoch 58/60
16/16 [==============================] - 2s 127ms/step - loss: 0.2641 - accuracy: 0.9213
Epoch 59/60
16/16 [==============================] - 2s 138ms/step - loss: 0.1408 - accuracy: 0.9563
Epoch 60/60
16/16 [==============================] - 2s 133ms/step - loss: 0.1821 - accuracy: 0.9399
```

```
score = model.evaluate(x_test,  y_test)

print('\nScore', score)
```

```
17/17 [==============================] - 0s 4ms/step - loss: 0.0255 - accuracy: 0.9981

Score [0.025546979159116745, 0.9980582594871521]
```



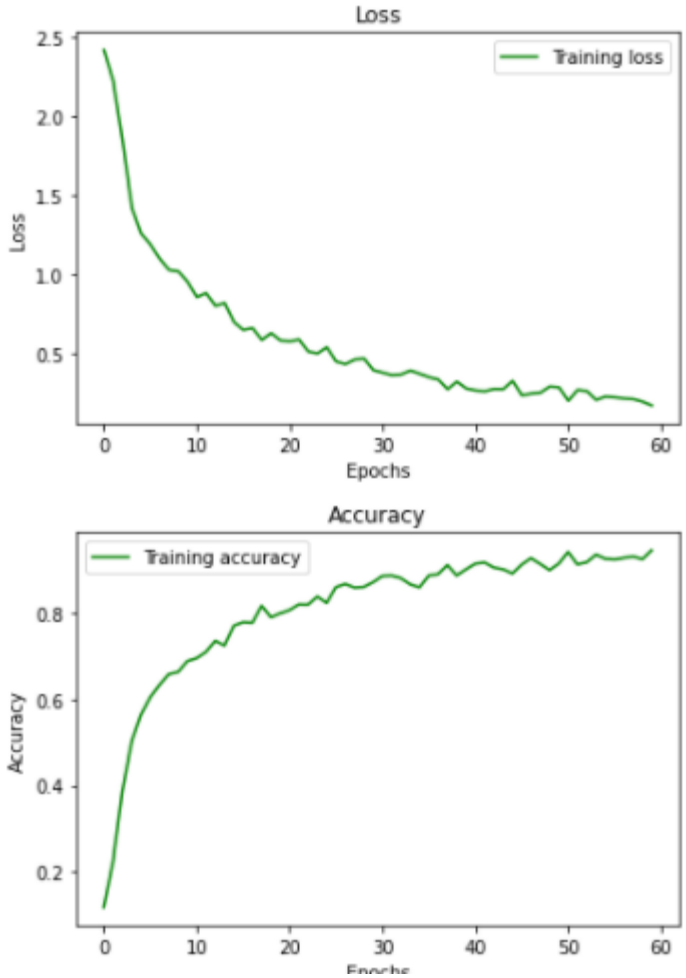
➔ Batch size = 64 , Συνελικτικά επίπεδα 128 και 64

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 98, 98, 128)       3584
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 49, 49, 128)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 47, 47, 64)        73792
_________________________________________________________________
activation_18 (Activation)   (None, 47, 47, 64)        0
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 23, 23, 64)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               4333696
_________________________________________________________________
activation_19 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_13 (Dense)             (None, 10)                1290
_________________________________________________________________
activation_20 (Activation)   (None, 10)                0
=================================================================
Total params: 4,412,362
Trainable params: 4,412,362
Non-trainable params: 0
_________________________________________________________________
```

```
16/16 [==============================] - 2s 142ms/step - loss: 0.2390 - accuracy: 0.9207
Epoch 57/60
16/16 [==============================] - 2s 132ms/step - loss: 0.2443 - accuracy: 0.9270
Epoch 58/60
16/16 [==============================] - 2s 136ms/step - loss: 0.2265 - accuracy: 0.9203
Epoch 59/60
16/16 [==============================] - 2s 139ms/step - loss: 0.1918 - accuracy: 0.9268
Epoch 60/60
16/16 [==============================] - 2s 140ms/step - loss: 0.1706 - accuracy: 0.9501
```

```
score = model.evaluate(x_test,  y_test)

print('\nScore', score)
```

```
17/17 [==============================] - 0s 5ms/step - loss: 0.0330 - accuracy: 0.9942

Score [0.03297937661409378, 0.9941747784614563]
```



Με την μείωση του batch size παρατηρείται μεγαλύτερη εμφάνιση του overfitting καθώς επίσης και τα τελικά αποτελέσματα έχουν μικρότερη ακρίβεια σε σχέση με προηγούμενες εκπαιδεύσεις.

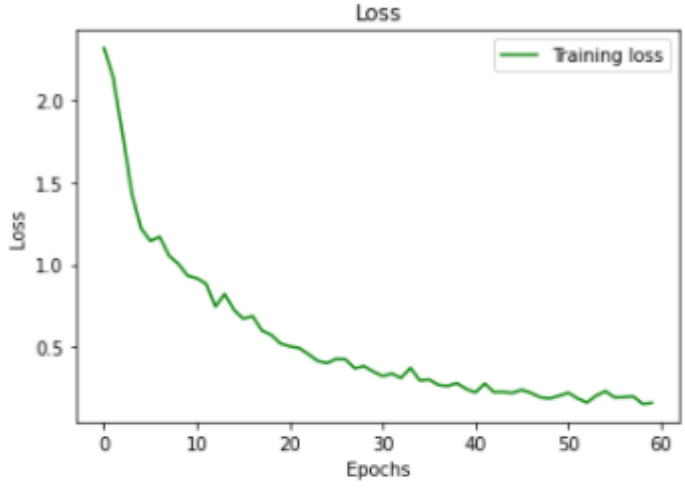
➔ Fully connected layers = 256

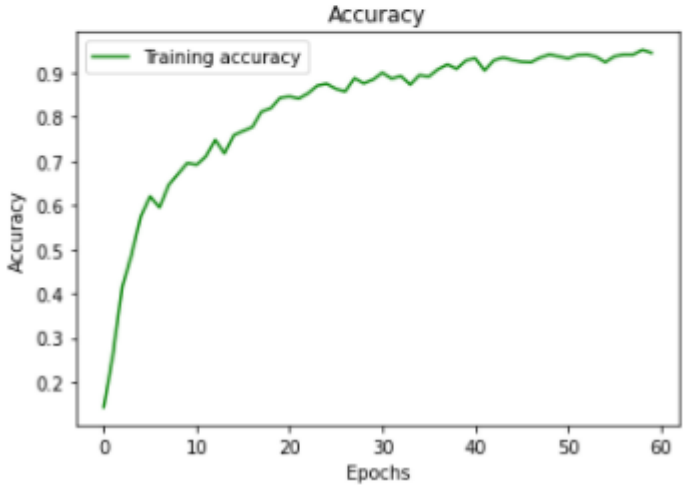
```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 98, 98, 64)        1792
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 47, 47, 32)        18464
_________________________________________________________________
activation_24 (Activation)   (None, 47, 47, 32)        0
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 23, 23, 32)        0
_________________________________________________________________
flatten_8 (Flatten)          (None, 16928)             0
_________________________________________________________________
dense_16 (Dense)             (None, 256)               4333824
_________________________________________________________________
activation_25 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_17 (Dense)             (None, 10)                2570
_________________________________________________________________
activation_26 (Activation)   (None, 10)                0
=================================================================
Total params: 4,356,650
Trainable params: 4,356,650
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 54/60
16/16 [==============================] - 4s 248ms/step - loss: 0.1283 - accuracy: 0.9571
Epoch 55/60
16/16 [==============================] - 4s 248ms/step - loss: 0.1155 - accuracy: 0.9684
Epoch 56/60
16/16 [==============================] - 4s 247ms/step - loss: 0.1040 - accuracy: 0.9675
Epoch 57/60
16/16 [==============================] - 4s 246ms/step - loss: 0.0992 - accuracy: 0.9683
Epoch 58/60
16/16 [==============================] - 4s 245ms/step - loss: 0.0820 - accuracy: 0.9768
Epoch 59/60
16/16 [==============================] - 4s 246ms/step - loss: 0.1105 - accuracy: 0.9646
Epoch 60/60
16/16 [==============================] - 4s 245ms/step - loss: 0.1376 - accuracy: 0.9555
```

```
score = model.evaluate(x_test,  y_test)

print('\nScore', score)
```

```
17/17 [==============================] - 0s 4ms/step - loss: 0.0255 - accuracy: 0.9981

Score [0.025546979159116745, 0.9980582594871521]
```





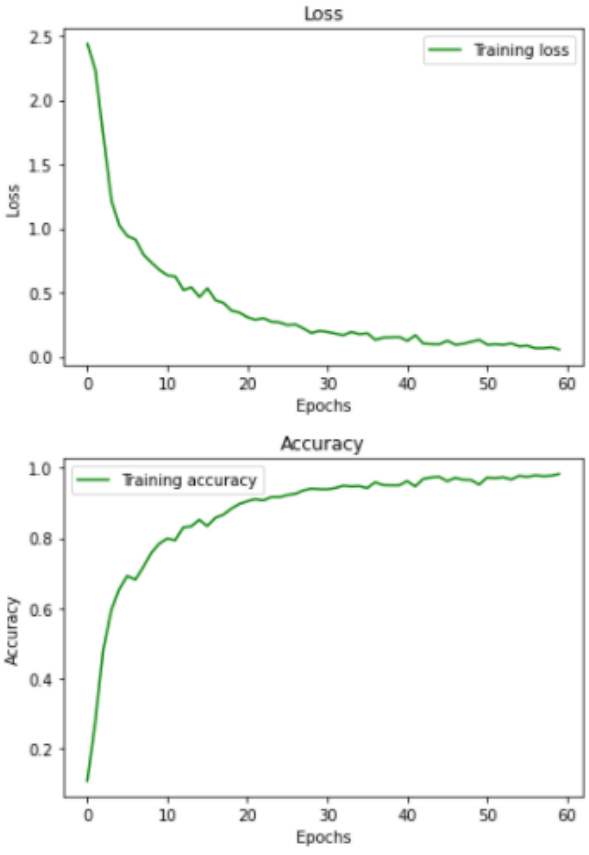
➔ Fully connected layers = 512

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 98, 98, 64)        1792
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 47, 47, 32)        18464
_________________________________________________________________
activation_27 (Activation)   (None, 47, 47, 32)        0
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 23, 23, 32)        0
_________________________________________________________________
flatten_9 (Flatten)          (None, 16928)             0
_________________________________________________________________
dense_18 (Dense)             (None, 512)               8667648
_________________________________________________________________
activation_28 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_19 (Dense)             (None, 10)                5130
_________________________________________________________________
activation_29 (Activation)   (None, 10)                0
=================================================================
Total params: 8,693,034
Trainable params: 8,693,034
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 53/60
16/16 [==============================] - 4s 246ms/step - loss: 0.0900 - accuracy: 0.9748
Epoch 54/60
16/16 [==============================] - 4s 246ms/step - loss: 0.0995 - accuracy: 0.9686
Epoch 55/60
16/16 [==============================] - 4s 262ms/step - loss: 0.0862 - accuracy: 0.9769
Epoch 56/60
16/16 [==============================] - 4s 251ms/step - loss: 0.0702 - accuracy: 0.9807
Epoch 57/60
16/16 [==============================] - 4s 245ms/step - loss: 0.0608 - accuracy: 0.9788
Epoch 58/60
16/16 [==============================] - 4s 250ms/step - loss: 0.0627 - accuracy: 0.9786
Epoch 59/60
16/16 [==============================] - 4s 248ms/step - loss: 0.0689 - accuracy: 0.9810
Epoch 60/60
16/16 [==============================] - 4s 249ms/step - loss: 0.0566 - accuracy: 0.9828
```

```
[103] score = model.evaluate(x_test,  y_test)

      print('\nScore', score)
```

```
17/17 [==============================] - 0s 4ms/step - loss: 0.0095 - accuracy: 0.9961

Score [0.009537205100059 51, 0.9961165189743042]
```



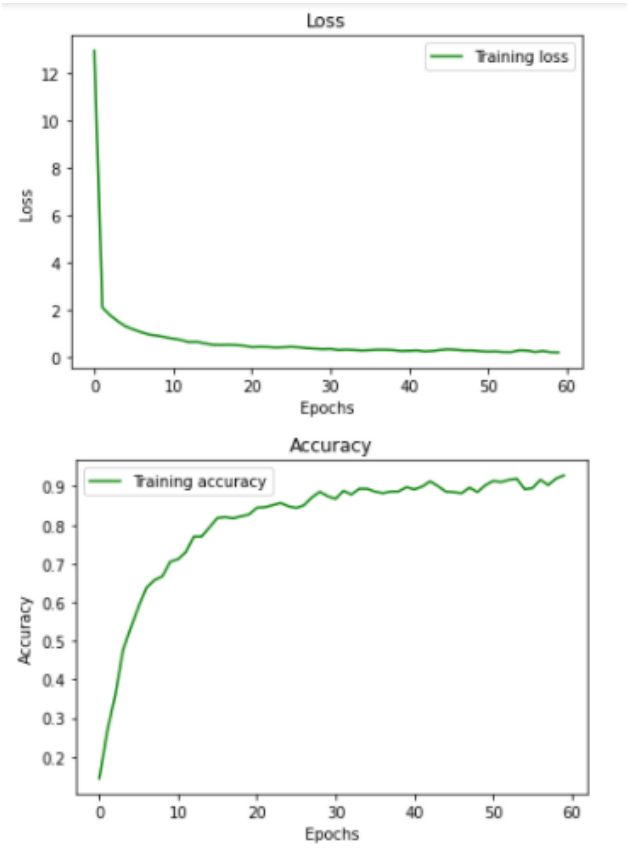
Με την τελευταία αρχιτεκτονική καταφέραμε να πετύχουμε το μεγαλύτερο ποσοστό ακρίβειας σε σχέση με όλα τα προηγούμενα αποτελέσματα. Επιπλέον, είχαμε και πολύ γενικευμένες απαντήσεις πράγμα που σημαίνει ότι υπάρχει ελάχιστη εμφάνιση overfitting καθώς είναι πιο ομαλή η έξοδος του accuracy και του loss κατά την εκπαίδευση.

- Ερώτημα d)

Σε αυτό το ερώτημα σύμφωνα και με την εκφώνηση χρησιμοποιούμε κατά τη δημιουργία του συνελικτικού νευρωνικού μας δικτύου κανονικοποίηση των δεδομένων μας με batch normalization. Αυτό το εφαρμόζουμε στο δικτυο που είχε την καλύτερη απόδοση από τα προηγούμενα που δημιουργήσαμε ώστε να δούμε πως εν τέλη θα επηρεαστεί η επίδοσή του.

Παρακάτω φαίνεται η αρχιτεκτονική του δικτύου μας καθώς επίσης και τα αποτελέσματα της εκπαίδευσης.

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_20 (Conv2D)           (None, 98, 98, 64)        1792
_________________________________________________________________
batch_normalization (BatchNo (None, 98, 98, 64)        256
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 47, 47, 32)        18464
_________________________________________________________________
batch_normalization_1 (Batch (None, 47, 47, 32)        128
_________________________________________________________________
activation_30 (Activation)   (None, 47, 47, 32)        0
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 23, 23, 32)        0
_________________________________________________________________
flatten_10 (Flatten)         (None, 16928)             0
_________________________________________________________________
dense_20 (Dense)             (None, 512)               8667648
_________________________________________________________________
activation_31 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_10 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_21 (Dense)             (None, 10)                5130
_________________________________________________________________
activation_32 (Activation)   (None, 10)                0
=================================================================
Total params: 8,693,418
Trainable params: 8,693,226
Non-trainable params: 192
_________________________________________________________________
```




```
Epoch 55/60
16/16 [==============================] - 4s 251ms/step - loss: 0.3529 - accuracy: 0.8791
Epoch 56/60
16/16 [==============================] - 4s 256ms/step - loss: 0.3145 - accuracy: 0.8942
Epoch 57/60
16/16 [==============================] - 4s 260ms/step - loss: 0.2543 - accuracy: 0.9117
Epoch 58/60
16/16 [==============================] - 4s 272ms/step - loss: 0.2783 - accuracy: 0.9022
Epoch 59/60
16/16 [==============================] - 4s 274ms/step - loss: 0.2550 - accuracy: 0.9179
Epoch 60/60
16/16 [==============================] - 4s 266ms/step - loss: 0.2162 - accuracy: 0.9353
```

```
score = model.evaluate(x_test,  y_test)

print('\nScore', score)
```

```
17/17 [==============================] - 0s 5ms/step - loss: 0.8206 - accuracy: 0.7612

Score [0.8206047415733337, 0.7611650228500366]
```

Εν τέλη, παρατηρείται σημαντική πτώση στη τελική ακρίβεια των αποτελεσμάτων που έχουμε καθώς επίσης και μεγαλύτερη εμφάνιση overfitting σε σχέση με τα προηγούμενα αποτελέσματα που είχαμε.

- Ερώτημα e)

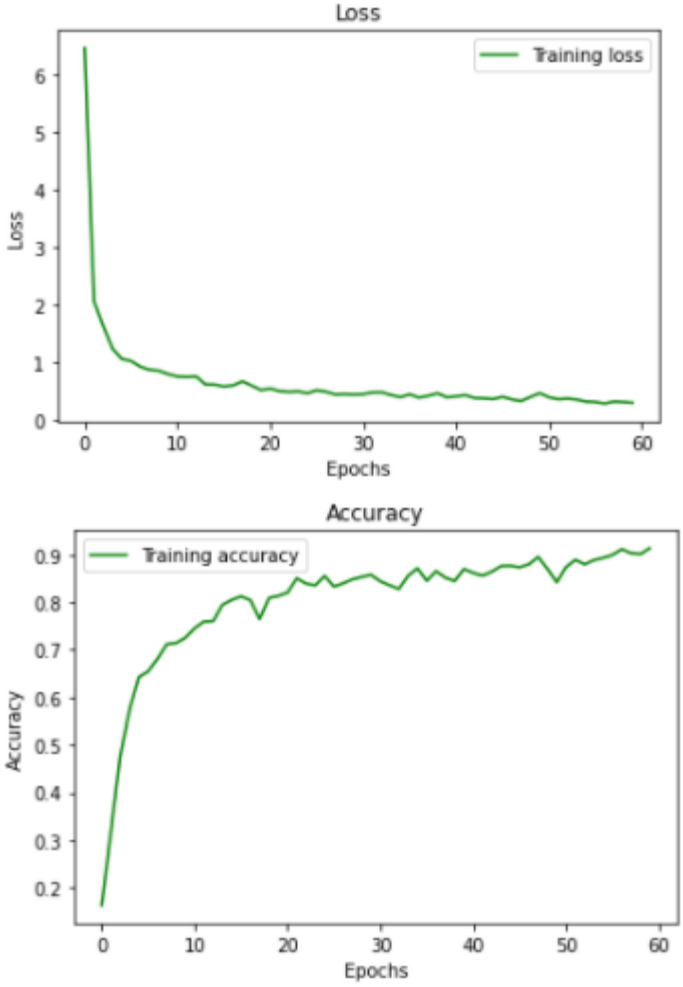
Γι αυτό το τελευταίο ερώτημα δημιουργούμε ένα συνελικτικό δίκτυο όπως και προηγουμένως αυτή τη φορά όμως χωρίς να χρησιμοποιούμε fully connected layer όπως επίσης και χωρίς pooling. Αυτή τη φορά εξετάσαμε την συγκεκριμένη εκδοχή του δικτύου στο αρχικό μας δικτυο ώστε να εξετάσουμε τα αποτελέσματα συγκριτικά με την αρχική εκπαίδευση. Παρακάτω φαίνονται τα αντίστοιχα αποτελέσματα.

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 98, 98, 64)        1792
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 96, 96, 32)        18464
_________________________________________________________________
activation_45 (Activation)   (None, 96, 96, 32)        0
_________________________________________________________________
flatten_15 (Flatten)         (None, 294912)            0
_________________________________________________________________
dense_30 (Dense)             (None, 10)                2949130
_________________________________________________________________
activation_46 (Activation)   (None, 10)                0
=================================================================
Total params: 2,969,386
Trainable params: 2,969,386
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 56/60
16/16 [==============================] - 4s 255ms/step - loss: 0.3067 - accuracy: 0.9026
Epoch 57/60
16/16 [==============================] - 4s 254ms/step - loss: 0.3175 - accuracy: 0.9060
Epoch 58/60
16/16 [==============================] - 4s 253ms/step - loss: 0.3255 - accuracy: 0.9095
Epoch 59/60
16/16 [==============================] - 4s 255ms/step - loss: 0.3087 - accuracy: 0.9053
Epoch 60/60
16/16 [==============================] - 4s 271ms/step - loss: 0.3266 - accuracy: 0.9042
```

```
score = model.evaluate(x_test,  y_test)

print('\nScore', score)
```

```
17/17 [==============================] - 0s 7ms/step - loss: 0.1116 - accuracy: 0.9709

Score [0.1115623414516449, 0.9708737730979919]
```



Εν τέλη παρατηρείται ότι τα αποτελέσματα δείχνουν να μην βελτιώνονται σε σχέση με αυτά τα οποία είχαμε αρχικά πράγμα το οποίο οφείλεται και στο γεγονός ότι αφαιρέθηκαν κάποια παραπάνω χαρακτηριστικά στην αρχιτεκτονική των δικτύων μας. Αυτό είχε και ως αποτέλεσμα το να αυξηθεί και το overfitting πάλι συγκριτικά με την πρώτη εκπαίδευση του συγκεκριμένου δικτύου από το πρώτο ερώτημα.

Η καλύτερη αρχιτεκτονική

Εν τέλη η αρχιτεκτονική με την καλύτερη απόδοση ήταν αυτή η οποία είχε 2 συνελικτικά επίπεδα με 64 και 32 φίλτρα ενώ ταυτόχρονα, είχε 512 νευρώνες στο fully connected layer. Επιπλέον, άλλο ένα χαρακτηριστικό της εκπαίδευσης ήταν ότι χρησιμοποιήθηκε batch size ισο με 128. Με βάση τα αποτελέσματα που είχαμε, κατα την εκπαίδευση επιτεύχθηκε ποσοστό ακρίβειας 98% ενώ αντίστοιχα για το test μετά την ολοκλήρωση της εκπαίδευσης είχαμε 99% ακρίβεια. Αυτό ήταν και το μοντέλο το οποίο παρά τα βέλτιστά του αποτελέσματα είναι και πολύ μικρότερη εμφάνιση overfitting.

Άσκηση 3η

Γενικότερα τα συνελικτικά νευρωνικά δίκτυα έχουν το πλεονέκτημα ότι χωρίς να απαιτούν περισσότερους πόρους έχουν ταυτόχρονα περισσότερες παραμέτρους πράγμα το οποίο βοηθάει στην ακρίβεια την οποία μπορούν να πετύχουν. Ετσι, τα ANN γίνονται αυτόματα λιγότερο αποτελεσματικά σε σχέση με τα CNN πράγμα που αναλύεται καλύτερα παρακάτω.

Για τον υπολογισμό των παραμέτρων στα ANN έχουμε Input * Output + Biases. Άρα αν είχαμε ένα δίκτυο με ένα layer που αποτελείται από 2 εισόδους και 7 εξόδους θα είχαμε 2*7+2, 16 παραμέτρους για το συγκεκριμένο δίκτυο. Από την άλλη, στα CNN έχουμε συνελικτικά επίπεδα τα οποία λαμβάνουν ως παράμετρο το μέγεθος του παραθύρου ενός φίλτρου, πράγμα το οποίο επηρεάζει και τον τελικό υπολογισμό των παραμέτρων. Επομένως αν για παράδειγμα είχαμε ένα layer με 32 εισόδους και 64 εξόδους και ένα φίλτρο με παράθυρο 3x3 τότε θα είχαμε 32*64*3*3+32 παραμέτρους. Άρα εύκολα βγαίνει το συμπέρασμα ότι τελικά τα συνελικτικά δίκτυα έχουν πολύ μεγαλύτερο αριθμό παραμέτρων πράγμα που επηρεάζει και σημαντικά το τελικό αποτέλεσμα της εκπαίδευσης. Αυτό επίσης τα καθιστά και πιο διαδεδομένα όσον αφορά τα προβλήματα αναγνώρισης εικόνας.