HW2 – ΤΗΛ 311

Μιχαήλ Κρατημένος

ΑΜ: 2018030104

Θέμα 1: Λογιστική Παλινδρόμηση: Αναλυτική εύρεση κλίσης (Gradient)

α) $h_\theta(x) = f(\theta^T x),\ f(z) = \frac{1}{1+e^{-z}}, \theta = [\theta_1, \theta_2, \dots, \theta_n]^T,$

$x = [x_1, x_2, \dots, x_n]^T$

- $\frac{d}{d\theta_j}\left(h_\theta(x)\right) = \frac{d}{d\theta_j}\left(\frac{1}{1+e^{-\theta^T x}}\right) = \frac{-1}{\left(1+e^{-\theta^T x}\right)^2} * \frac{d}{d\theta_j}\left(1 + e^{-\theta^T x}\right) =$

  $$\frac{-1}{\left(1 + e^{-\theta^T x}\right)^2} * \frac{d}{d\theta_j}\left(e^{-\theta^T x}\right) = \frac{-e^{-\theta^T x}}{\left(1 + e^{-\theta^T x}\right)^2} * \frac{d}{d\theta_j}(-\theta^T x) =$$

  $$\frac{e^{-\theta^T x}}{\left(1 + e^{-\theta^T x}\right)^2} * \frac{d}{d\theta_j}\left(\sum_{k=1}^{n} \theta_k * x_k\right) = \frac{x_j * e^{-\theta^T x}}{\left(1 + e^{-\theta^T x}\right)^2}\ (1)$$

- $1 - h_\theta(x) = 1 - \frac{1}{1+e^{-\theta^T x}} = \frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}(2)$
- $J(\theta) = \frac{1}{m} * \sum_{i=1}^{m}(-y^{(i)} * \ln\left(h_\theta(x^{(i)})\right) -$
  $(1 - y^{(i)}) * \ln\left(1 - h_\theta(x^{(i)})\right)$
- $\frac{dJ(\theta)}{d\theta_j} = \frac{1}{m} * \sum_{i=1}^{m}(-y^{(i)} * \frac{1}{h_\theta(x^{(i)})} * \frac{d}{d\theta_j}\left(h_\theta(x^{(i)})\right) +$

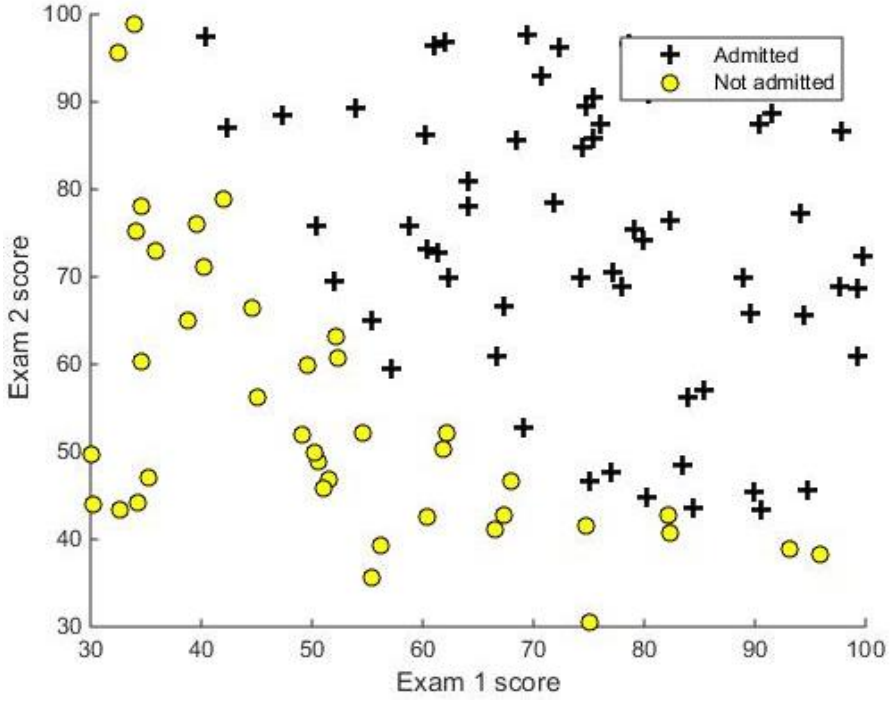
  $\left(1 - y^{(i)}\right) * \frac{1}{1-h_\theta(x^{(i)})} * \frac{d}{d\theta_j}\left(h_\theta\left(x^{(i)}\right)\right) \overset{(1)}{(2)}=$

  $$\frac{1}{m} * \sum_{i=1}^{m}\left((-y^{(i)} * \frac{{x_j}^{(i)} * e^{-\theta^T x^{(i)}}}{\left(1 + e^{-\theta^T x^{(i)}}\right)}\right) + \left(1 - y^{(i)}\right) * \frac{{x_j}^{(i)}}{\left(1 + e^{-\theta^T x^{(i)}}\right)})$$

  $$= \frac{1}{m} * \sum_{i=1}^{m}(-y^{(i)} * {x_j}^{(i)} * \left(1 - h_\theta\left(x^{(i)}\right)\right)$$

  $$+\left(1 - y^{(i)}\right) * {x_j}^{(i)} * h_\theta\left(x^{(i)}\right) = \frac{1}{m} * \sum_{i=1}^{m} {x_j}^{(i)} * \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

β) Εφαρμόστηκε λογιστική παλινδρόμηση σε ένα dataset για να προβλεφθεί αν ένας φοιτητής θα γίνει δεκτός σε ένα πανεπιστήμιο με βάση τους βαθμούς του σε δύο εξετάσεις, ως εξής:

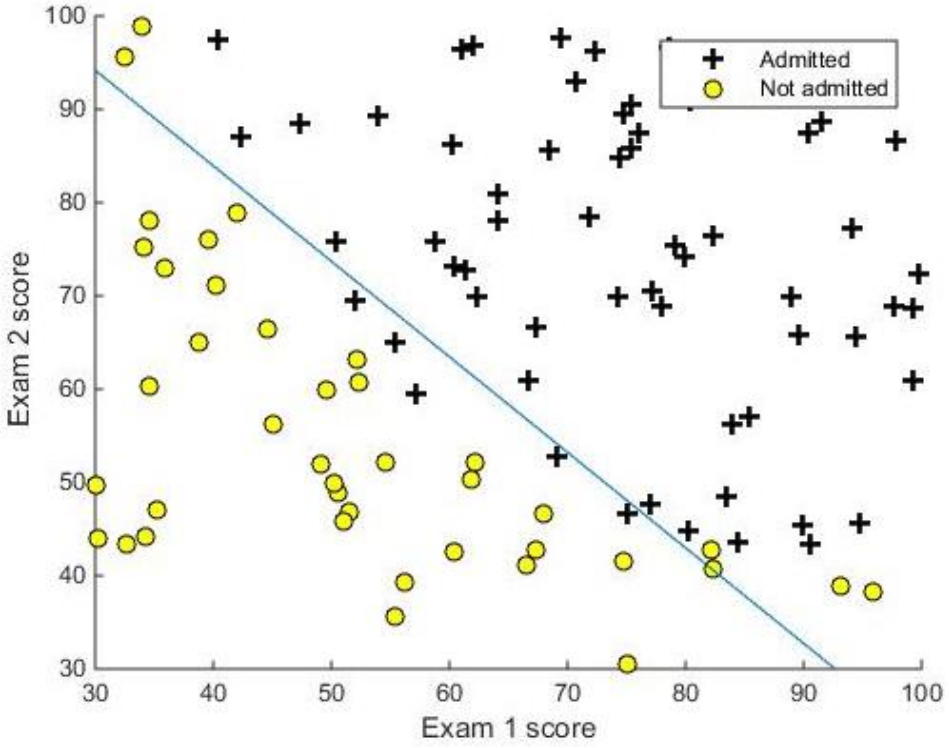


Μετά υπολογίζονται το κόστος και το gradient για τη λογιστική παλινδρόμηση μέσω της συνάρτησης costFunction και προκύπτουν τα εξής αποτελέσματα, τα οποία συμφωνούν με την εκφώνηση:

```
Cost at initial theta (zeros): 0.693147
Gradient at initial theta (zeros):
 -0.100000
 -12.009217
 -11.262842
```

Ύστερα, σχεδιάζεται η ευθεία παλινδρόμησης μέσω της ελαχιστοποίησης της συνάρτησης κόστους (cross-entropy) και αφού υπολογιστούν οι παράμετροι θ με τη χρήση της fminuc, ως εξής:

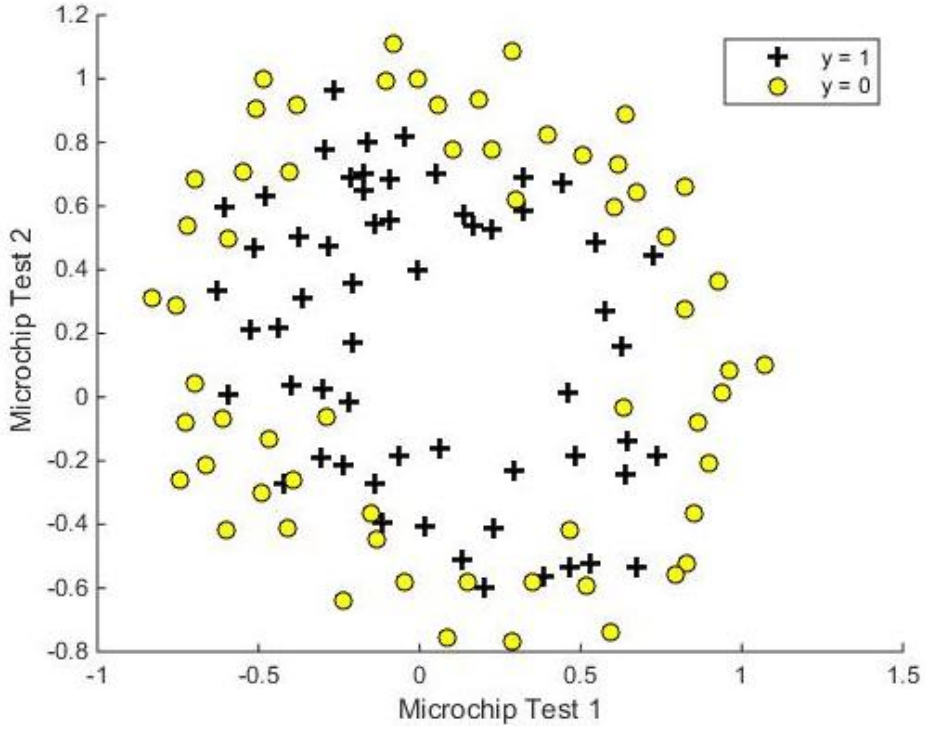
```
Cost at theta found by fminunc: 0.203506
theta:
 -24.932955
 0.204407
 0.199618
```



Τέλος, γίνεται η πρόβλεψη για την αποδοχή ενός φοιτητή με βαθμούς 45 και 85 και πιθανότητα αποδοχής 0.774323, με τη χρήση του θ και της σιγμωιδούς συνάρτησης καθώς και υπολογίζεται η ακρίβεια εκπαίδευσης στο 89% με τη χρήση της συνάρτησης predict.

Θέμα 2: Λογιστική Παλινδρόμηση με Ομαλοποίηση

a), b), c) Εφαρμόστηκε λογιστική παλινδρόμηση με ομαλοποίηση σε ένα dataset με δύο microchip tests, έτσι ώστε να προβλεφθεί αν τα μικροτσίπ από μια μονάδα κατασκευής περνούν τον έλεγχο ποιότητας (Quality Analysis)



Τα δεδομένα δεν μπορούν να διαχωριστούν με μια ευθεία γραμμή όπως στο προηγούμενο θέμα, οπότε γίνεται χρήση μιας παραλλαγής της λογιστικής παλινδρόμησης. Ειδικότερα, χρησιμοποιούνται μεγαλύτερες διαστάσεις για να υπολογιστεί ορθά το σύνορο απόφασης. Σε αυτήν την κατεύθυνση, προβάλλονται στο χώρο τα δεδομένα που ορίζουν όλοι οι όροι των x_1, x_2 μέχρι και $6^{ου}$ βαθμού με τη χρήση των σχέσεων που δόθηκαν στην εκφώνηση.

d) $J(\theta) = \frac{1}{m} * \sum_{i=1}^{m}(-y^{(i)} * \ln(h_\theta(x^{(i)})) -$

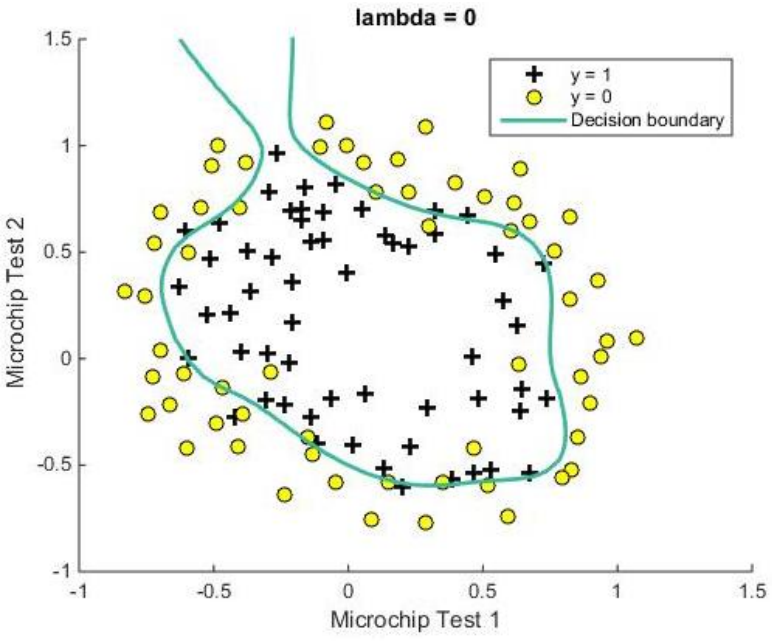
$$\left(1 - y^{(i)}\right) * \ln\left(1 - h_\theta\left(x^{(i)}\right)\right) + \frac{\lambda}{2m} * \sum_{j=1}^{m} {\theta_j}^2$$

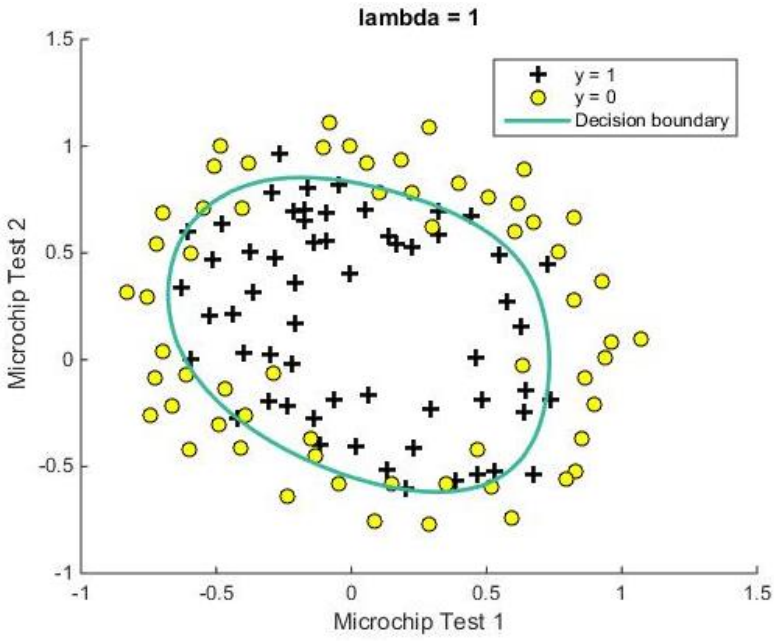
$$\frac{dJ(\theta)}{d\theta_j} = \frac{1}{m} * \frac{d}{d\theta_j}[\sum_{i=1}^{m}(-y^{(i)} * \ln(h_\theta(x^{(i)})) -$$

$$\left(1 - y^{(i)}\right) * \ln\left(1 - h_\theta\left(x^{(i)}\right)\right)] + \frac{d}{d\theta_j}(\frac{\lambda}{2m} * \sum_{j=1}^{m} {\theta_j}^2) \overset{\theta\acute{\epsilon}\mu\alpha\ 1}{\Longleftrightarrow}$$

$$\frac{dJ(\theta)}{d\theta_j} = \frac{1}{m} * \sum_{i=1}^{m}({x_j}^{(i)} * (h_\theta(x^{(i)}) - y^{(i)})) + \frac{\lambda}{m} * \theta_j$$

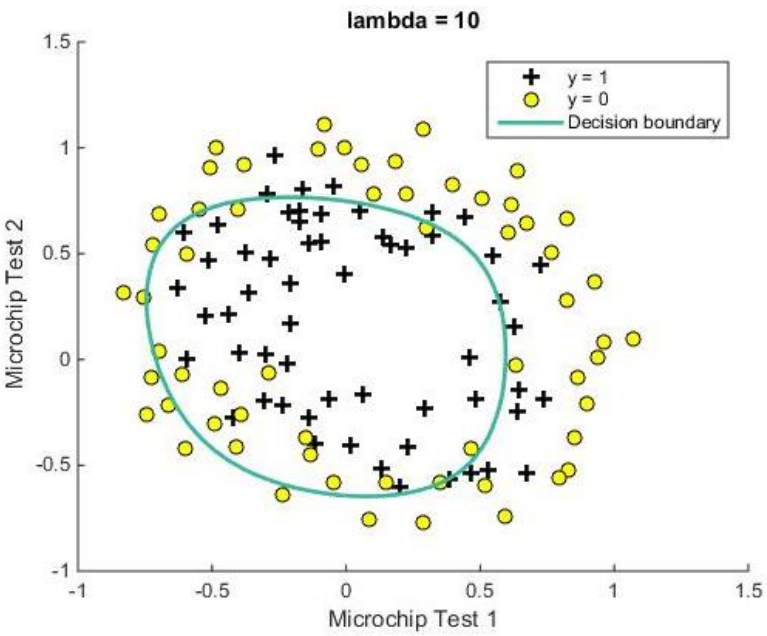
e), f) Αφού βελτιστοποιήθηκαν οι παράμετροι, σχεδιάστηκαν τα σύνορα απόφασης για διαφορετικές τιμές της παραμέτρου λ ως εξής:

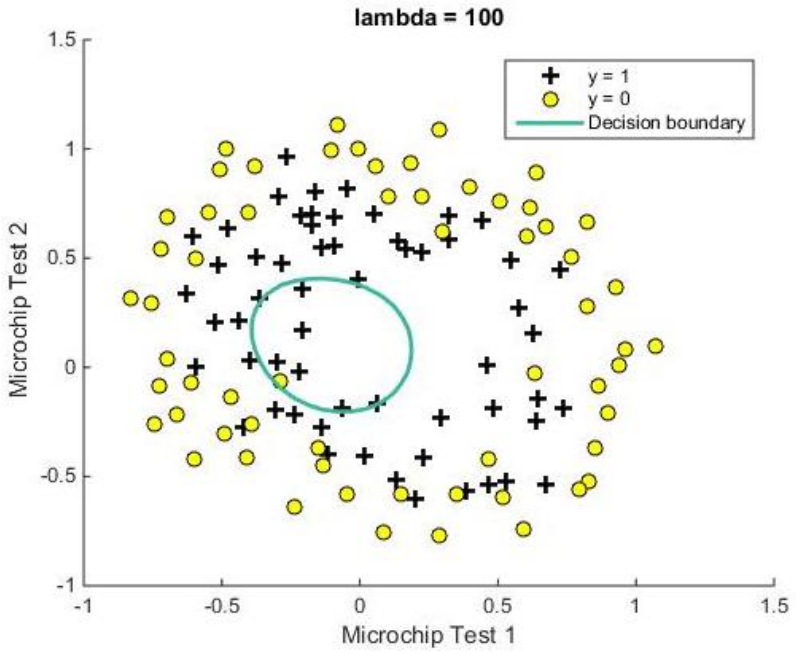




Accuracy = 87.29%

Accuracy = 84.75%





Accuracy = 72.03%

Accuracy = 55.08%

- Παρατηρείται ότι για λ=0, το μοντέλο πετυχαίνει τη μεγαλύτερη ακρίβεια, ωστόσο φαίνεται ότι το όριο απόφασης είναι αρκετά πολύπλοκο, δηλαδή γίνεται overfitting, κάτι το οποίο σημαίνει ότι το μοντέλο γενικεύει λάθος δεδομένα που βρίσκονται εκτός του training dataset.
- Για λ=100 από την άλλη, το μοντέλο πετυχαίνει τη μικρότερη ακρίβεια, ενώ παρατηρείται φαινόμενο underfitting, αφού δεν μοντελοποιείται καθόλου σωστά το training dataset. Άρα δεν υπάρχει καμία πιθανότητα γενίκευσης.
- Για λ=1 και λ=10 παρατηρείται ότι το όριο απόφασης είναι αρκετά γενικό, με σχετικά καλή ακρίβεια εντός του training dataset, άρα και με καλή γενίκευση σε δεδομένα που βρίσκονται εκτός του dataset αυτού.

Θέμα 3: Εκτίμηση Παραμέτρων (Maximum Likelihood)

Εύρεση εκτιμητή μέγιστης πιθανοφάνειας της παραμέτρου λ με MLE:

$$L_D(\lambda) = P(D|\lambda) = \prod_{k=1}^{n} p(x_k|\lambda), \qquad \lambda_{MLE} = argmax(P(D|\lambda))$$

$$l_D(\lambda) = \ln\big(L_D(\lambda)\big) = \sum_{k=1}^{n} \ln\left(p(x_k|\lambda)\right)$$

Πρέπει να ισχύει ότι: $\frac{d}{d\lambda} l_D(\lambda) = 0 \Rightarrow$

$$\frac{d}{d\lambda}\left[\sum_{k=1}^{n} \ln\big(p(x_k|\lambda)\big)\right] = \sum_{k=1}^{n} \frac{d}{d\lambda}\left[\ln\left(\frac{\lambda^{x_k} * e^{-\lambda}}{x_k!}\right)\right]$$

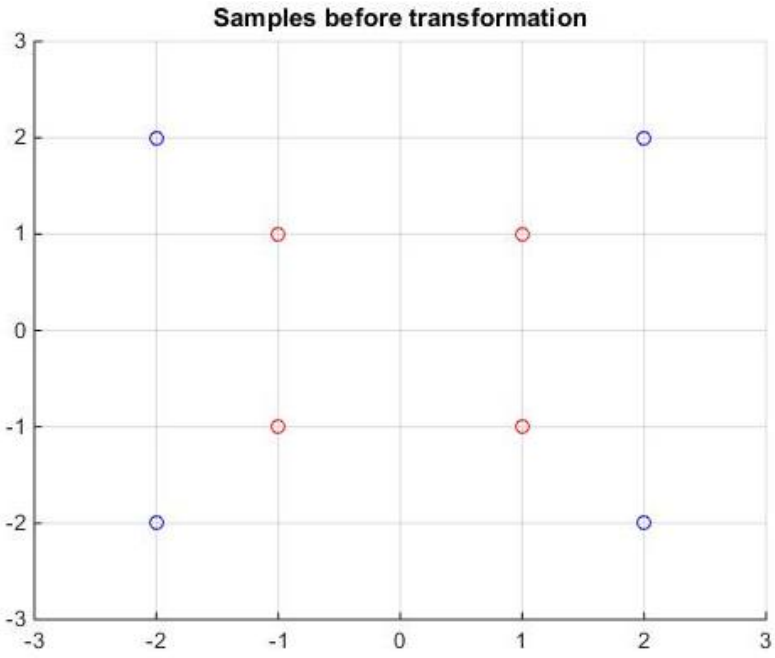
$$= \sum_{k=1}^{n}\left[\frac{x_k!}{\lambda^{x_k} * e^{-\lambda}} * \left(\frac{x_k * \lambda^{x_k-1} * e^{-\lambda} - \lambda^{x_k} * e^{-\lambda}}{x_k!}\right)\right]$$

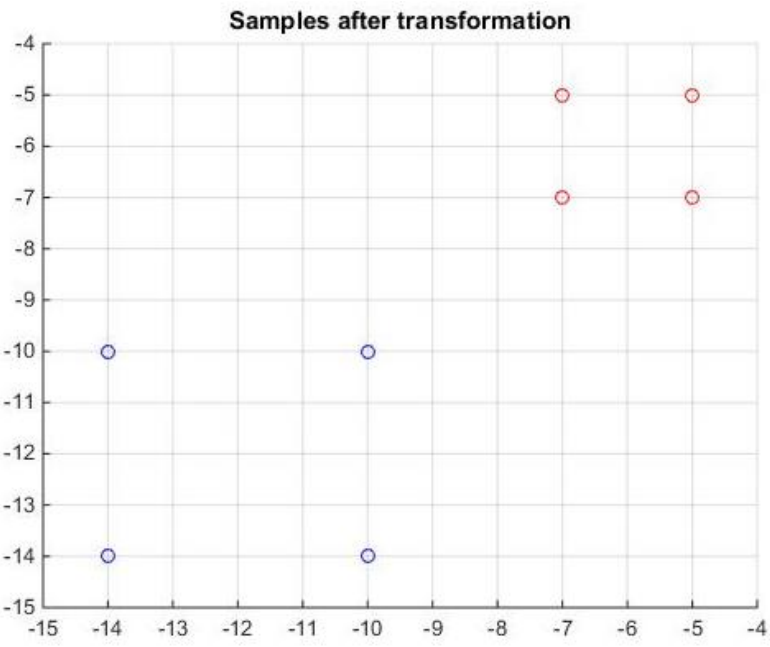
$$= \sum_{k=1}^{n}\left(\frac{\lambda^{x_k} * e^{-\lambda} * \left(\frac{x_k}{\lambda} - 1\right)}{\lambda^{x_k} * e^{-\lambda}}\right) = 0 \Rightarrow \frac{1}{\lambda} * \sum_{k=1}^{n} x_k = n$$

$$\Rightarrow \lambda = \frac{1}{n} * \sum_{k=1}^{n} x_k = \lambda_{MLE}$$

Θέμα 4: Support Vector Machines (Αναλυτική βελτιστοποίηση με ΚΚΤ)

Τα δείγματα πριν και μετά το μετασχηματισμό είναι τα εξής:





Το γραμμικό επίπεδο σχεδιασμού υπολογίζεται με αναλυτική βελτιστοποίηση ως εξής:

$$L(\omega, \omega_0, \lambda) = \frac{\omega^T * \omega}{2} - \sum_{i=1}^{N} \lambda_i * [y_i * (\omega^T * x_i + \omega_0) - 1] =$$

$$= \frac{{\omega_1}^2 + {\omega_2}^2}{2} - \lambda_1 * (-(-10\omega_1 - 10\omega_2 + \omega_0) - 1) -$$

$$\lambda_2 * (-7\omega_1 - 7\omega_2 + \omega_0 - 1)$$

Για την παράγωγο $(\frac{d}{d\omega} L(\omega, \omega_0, \lambda))$ ισχύει ότι:

- $\frac{d}{d\omega_1} L(\omega, \omega_0, \lambda) = 0 => \frac{2\omega_1}{2} - 10\lambda_1 + 7\lambda_2 = 0 =>$
  $\omega_1 = 10\lambda_1 - 7\lambda_2$
- $\frac{d}{d\omega_2} L(\omega, \omega_0, \lambda) = 0 => \frac{2\omega_2}{2} - 10\lambda_1 + 7\lambda_2 = 0 =>$
  $\omega_2 = 10\lambda_1 - 7\lambda_2$

Άρα $\omega_1 = \omega_2$

Έχουμε 2 SVM, άρα:

- $\lambda_1 * \left(- \left((\omega_1 \ \omega_2) * \begin{pmatrix} -10 \\ -10 \end{pmatrix} + \omega_0 \right) - 1 \right) = 0 \overset{\lambda_1 \neq 0}{\Longrightarrow}$

$10\omega_1 + 10\omega_2 - \omega_0 - 1 = 0 \overset{\omega_1 = \omega_2 = \omega}{\Longrightarrow} 20\omega - \omega_0 - 1 = 0 \ (1)$

- $\lambda_2 * \left((\omega_1 \ \omega_2) * \begin{pmatrix} -7 \\ -7 \end{pmatrix} + \omega_0 - 1 \right) = 0 \overset{\lambda_2 \neq 0}{\Longrightarrow}$

$-7\omega_1 - 7\omega_2 + \omega_0 - 1 = 0 \overset{\omega_1 = \omega_2 = \omega}{\Longrightarrow} -14\omega + \omega_0 - 1 = 0 \ (2)$

- $(1) + (2) => 6\omega - 2 = 0 => \omega = \frac{1}{3} = \omega_1 = \omega_2 (3)$
- $(1) \overset{(3)}{\Rightarrow} \frac{20}{3} - \omega_0 - 1 = 0 => \omega_0 = \frac{17}{3}$

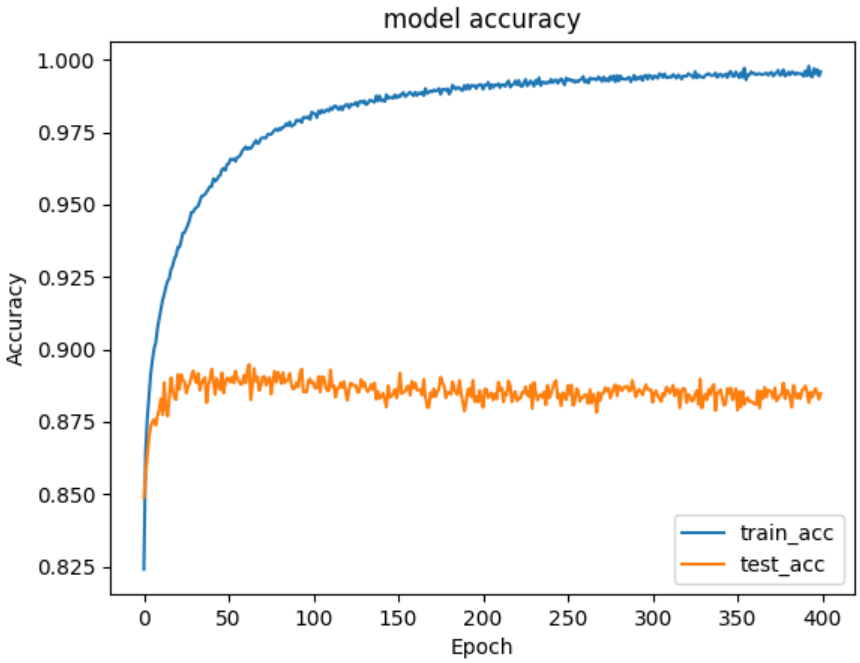
Άρα $\omega^T * x + \omega_0 = 0 => \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{17}{3} = 0 => \frac{x_1}{3} + \frac{x_2}{3} = -\frac{17}{3}$

$=> x_1 = -x_2 - 17$

Θέμα 6: Convolutional Neural Networks for Image Recognition

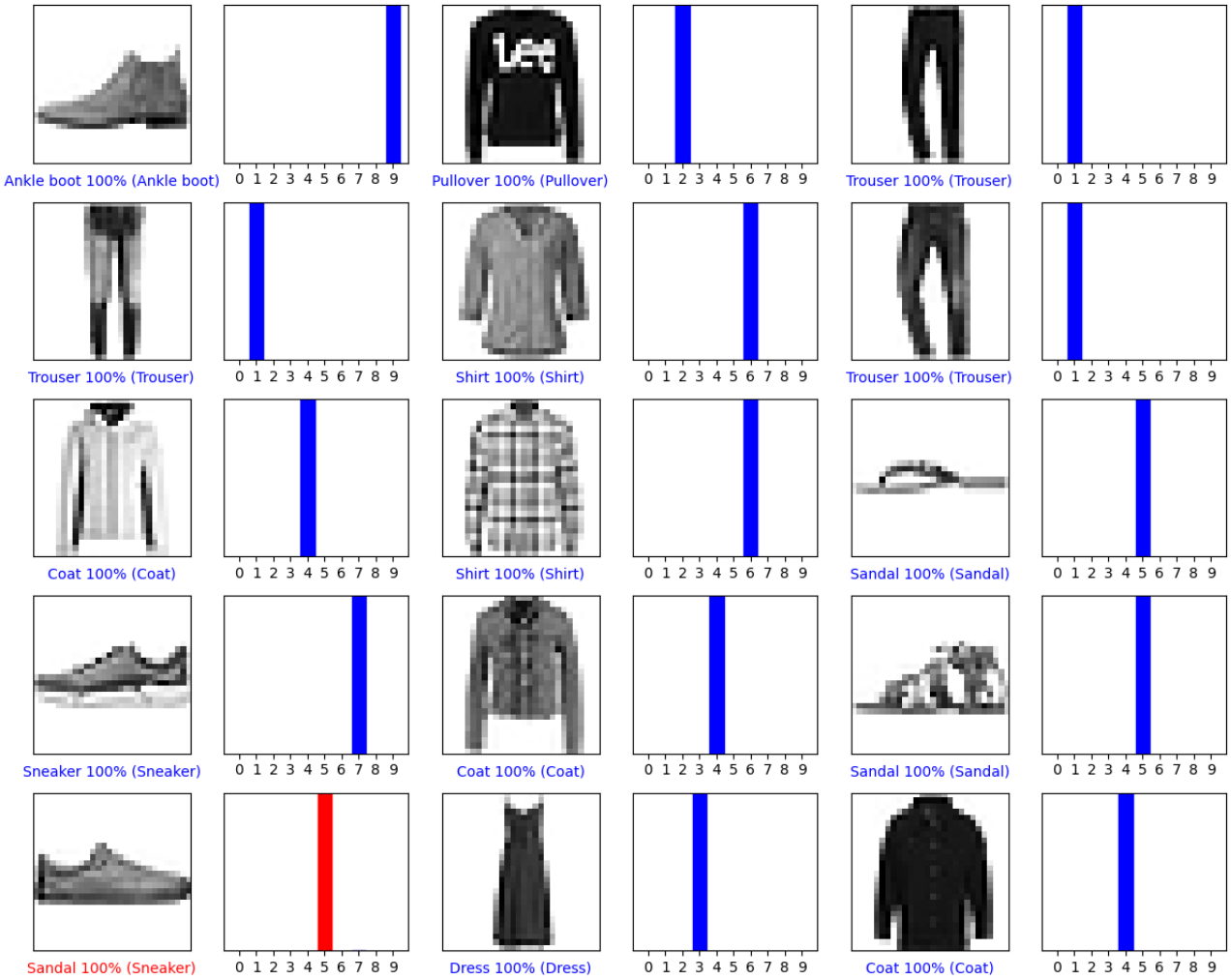
Οι κώδικες fashion, fashion_cnn έχουν περαστεί σε δύο αρχεία τύπου google colabs (ex_2_6_fashion και ex_2_6_fashion_cnn αντίστοιχα) γιατί χρειαζόταν να γίνει χρήση της GPU για να τρέχουν πιο γρήγορα οι κώδικες και για να αποθηκεύονται τα αποτελέσματα για τα διαφορετικά νήματα κώδικα που γράφονται για κάθε διαφορετική περίπτωση.

Αρχικά υλοποιείται ένας ταξινομητής με dense neural networks και παρακάτω φαίνονται τα γραφήματα με τις τιμές ακρίβειας εκπαίδευσης καθώς και με τις τιμές ακρίβειας επικύρωσης, για 400 epochs και με adam optimizer:



Παρατηρείται ότι το training accurancy τείνει στο 100%, ενώ το validate accurancy πλησιάζει το 90%.

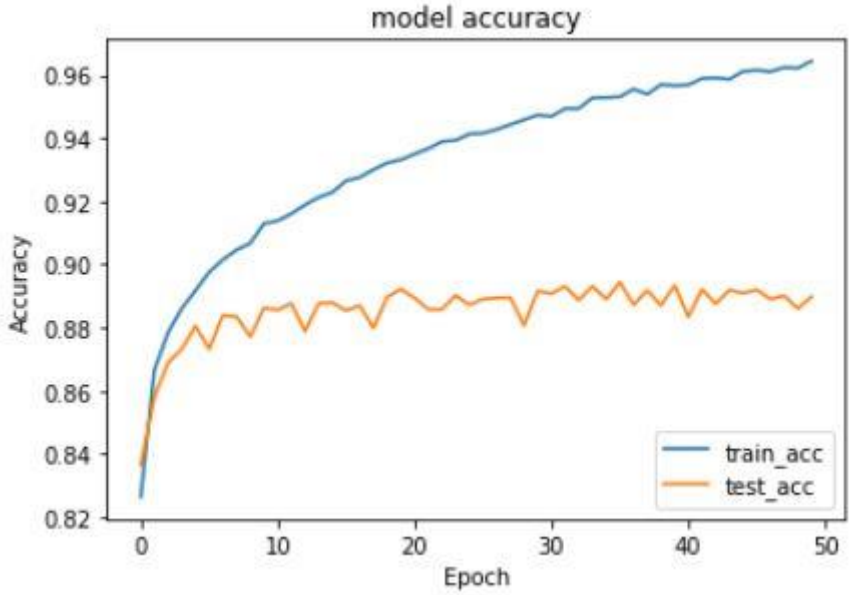
Παρακάτω φαίνονται οι προβλέψεις αναγνώρισης των προτύπων του μοντέλου:

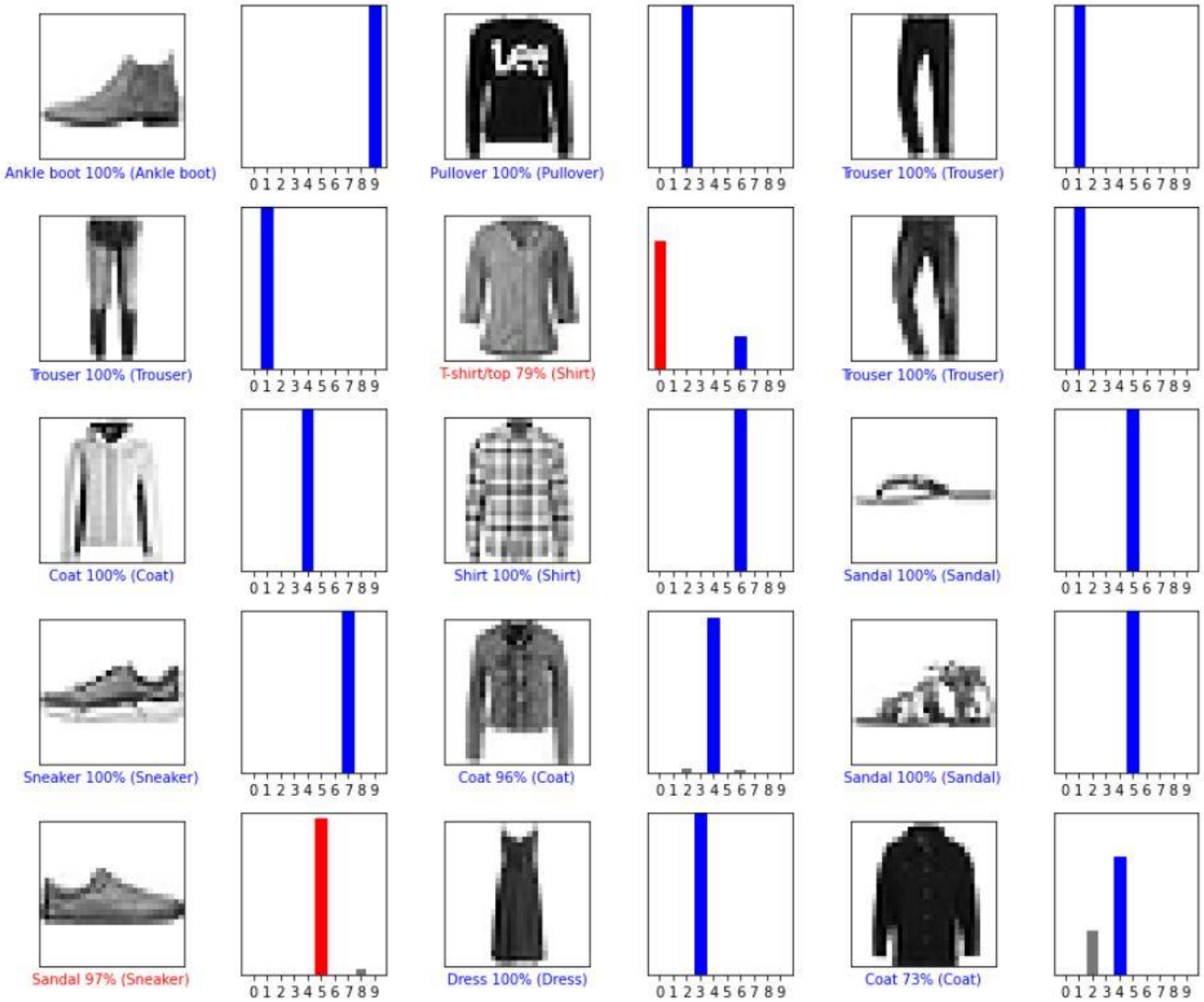


Ύστερα, δοκιμάστηκαν οι αλγόριθμοι βελτιστοποίησης, έτσι ώστε να βρεθεί αυτός που δίνει το καλύτερο accuracy. Τα epochs από 400 γίνονται 50 για να τρέχουν πιο γρήγορα τα προγράμματα, αφού μας ενδιαφέρει να συγκρίνουμε τους αλγορίθμους και όχι να βρούμε τη βέλτιστη εκδοχή τους.

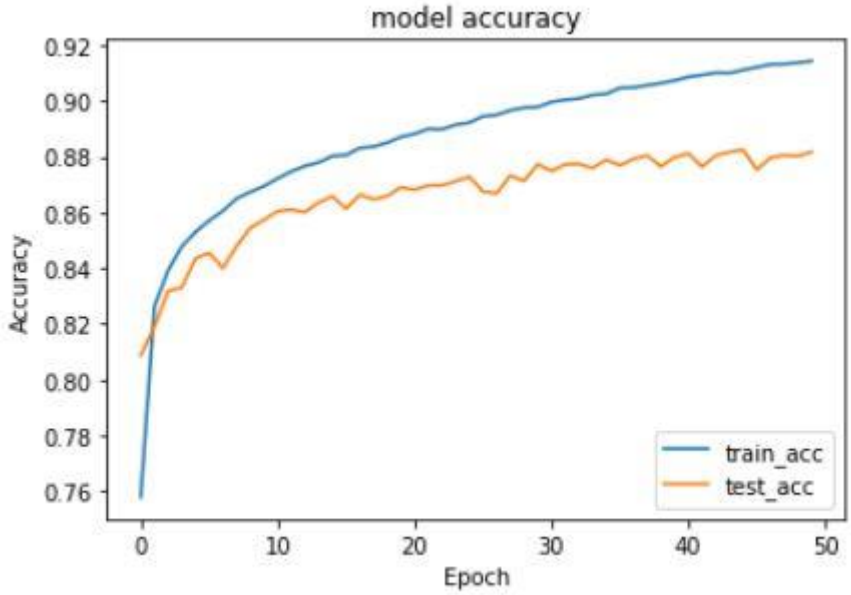
Παρατίθενται τα γραφήματα με τις τιμές ακρίβειας εκπαίδευσης καθώς και με τις τιμές ακρίβειας επικύρωσης, αλλά και οι προβλέψεις αναγνώρισης των προτύπων του μοντέλου για κάθε αλγόριθμο βελτιστοποίησης ξεχωριστά:

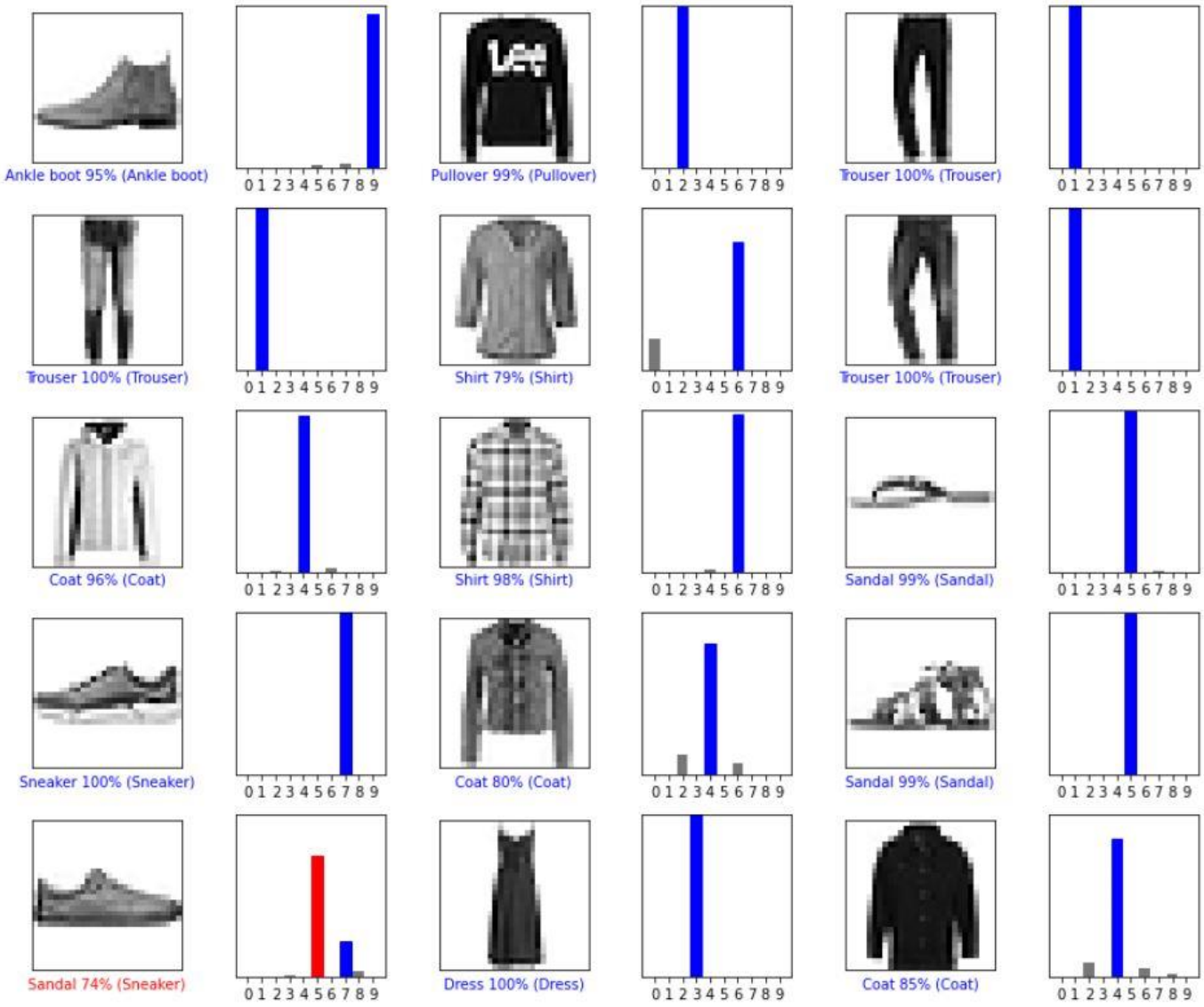
Adam optimizer:



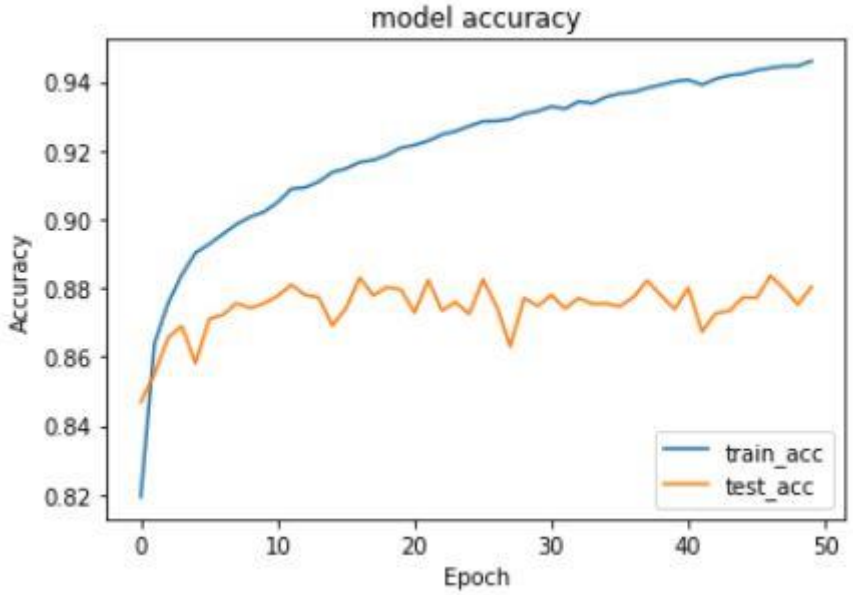


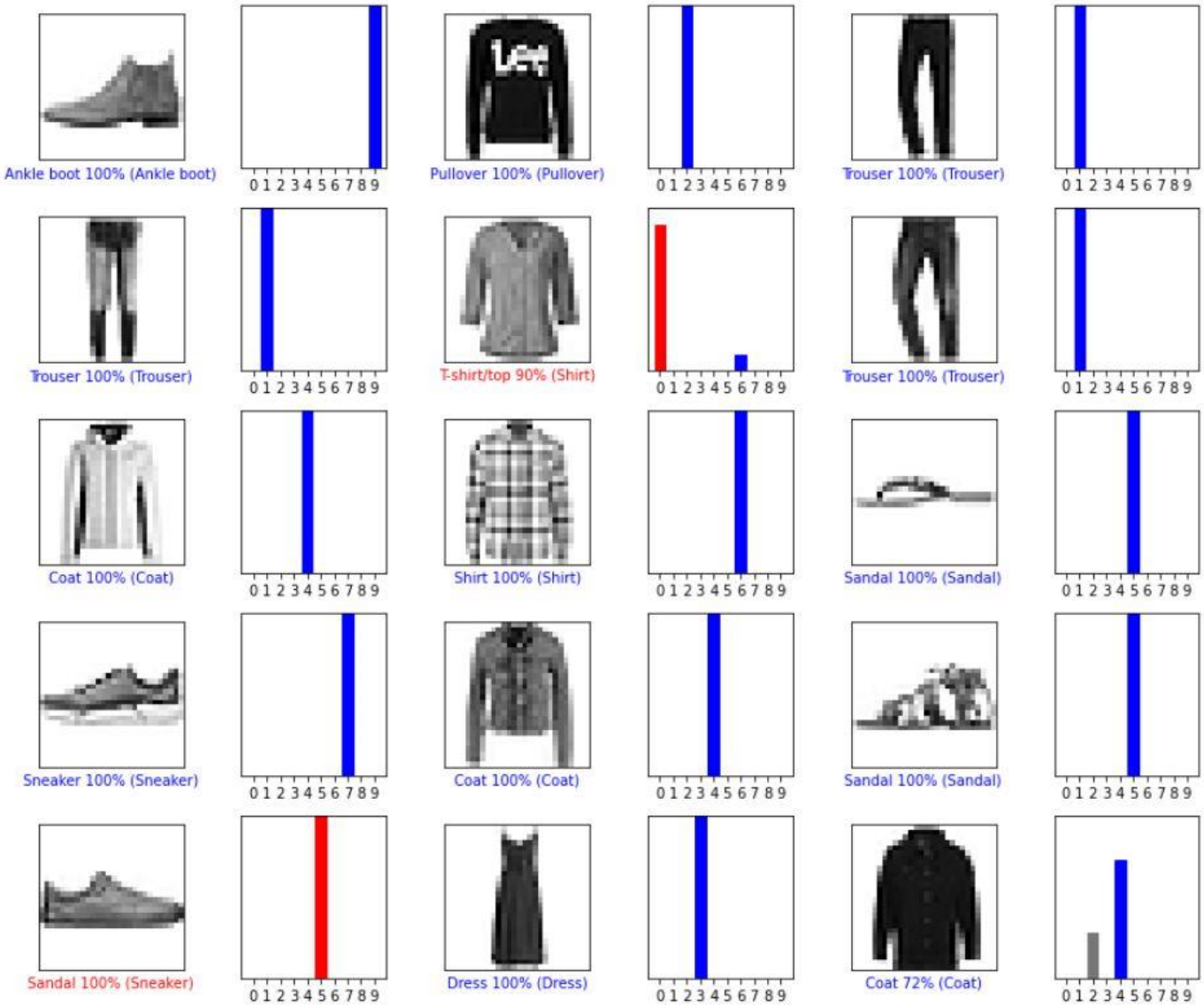
Sgd optimizer:



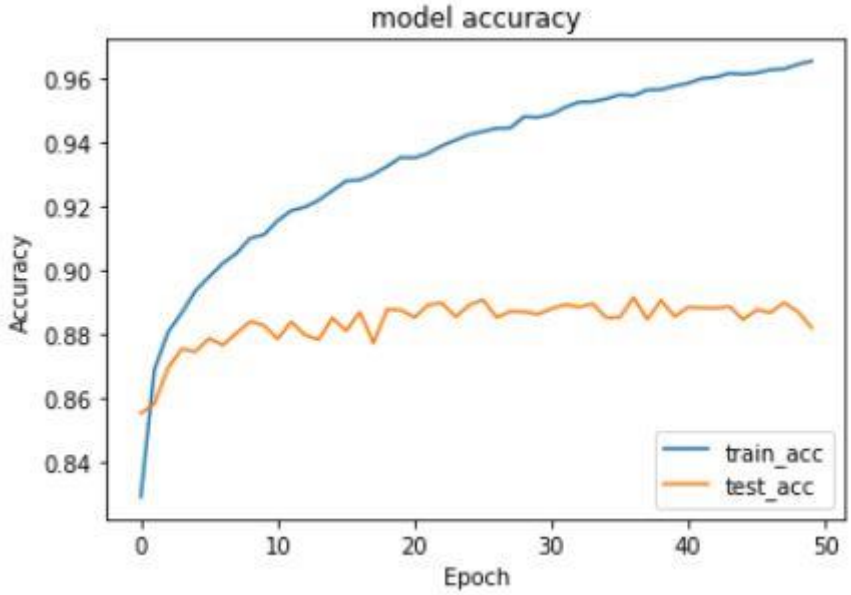


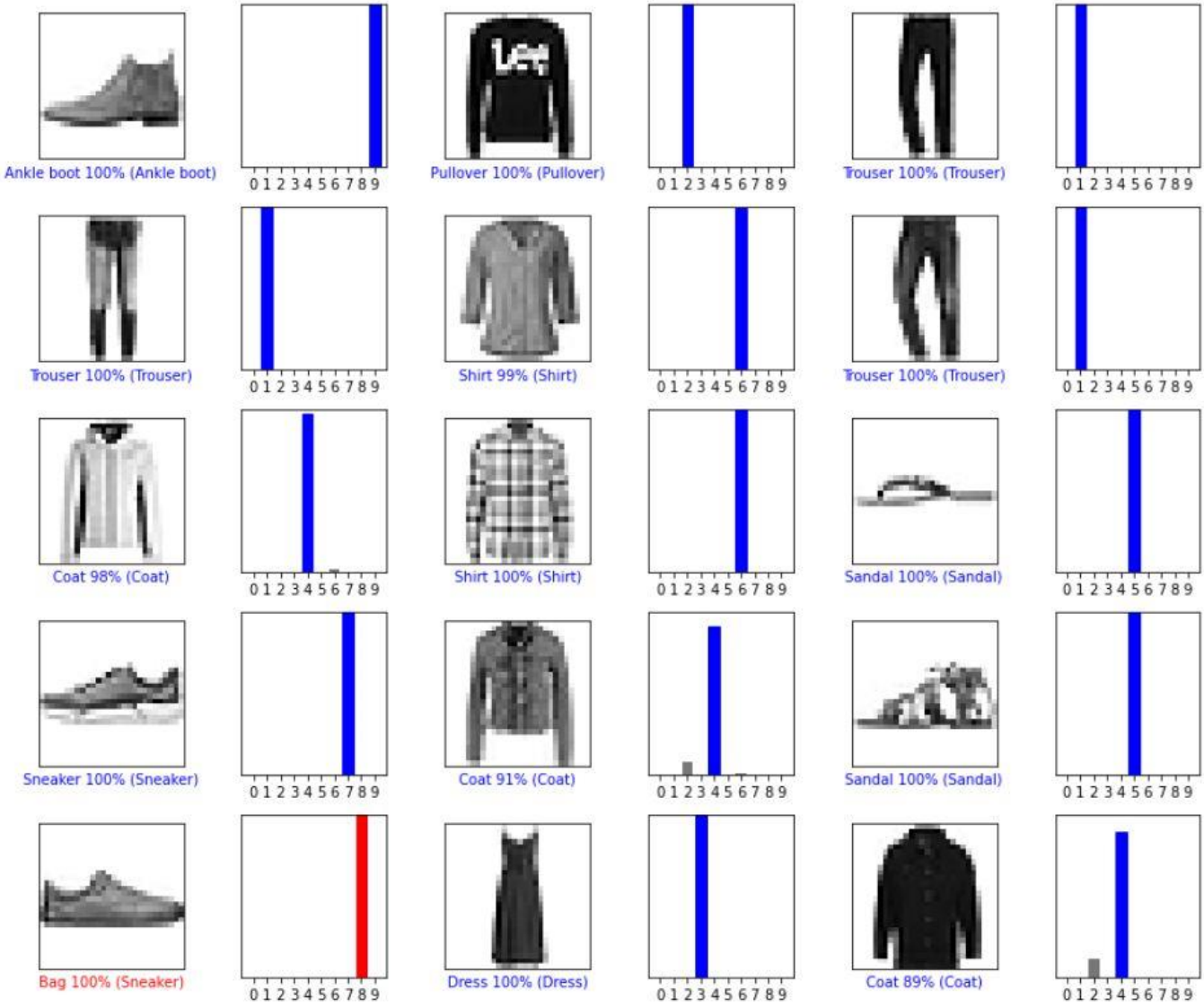
Rmsprop optimizer:



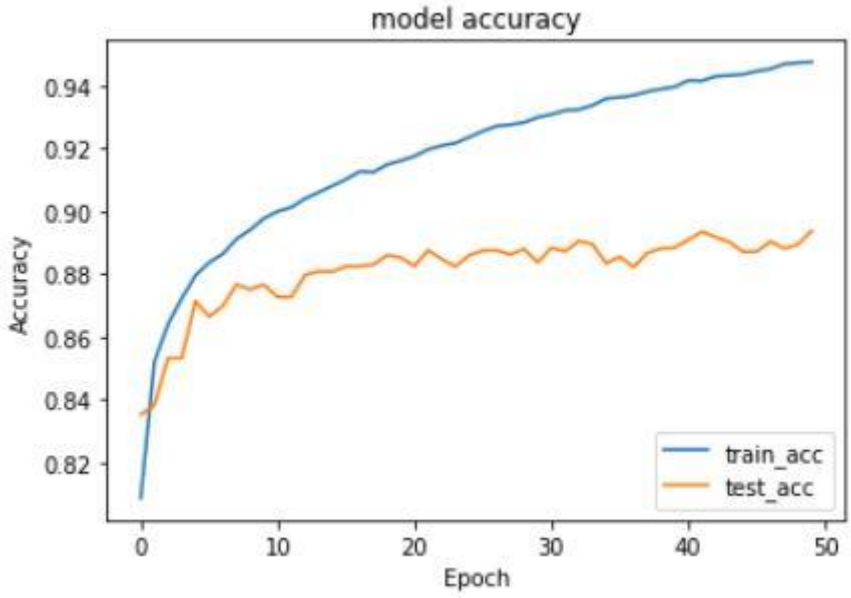


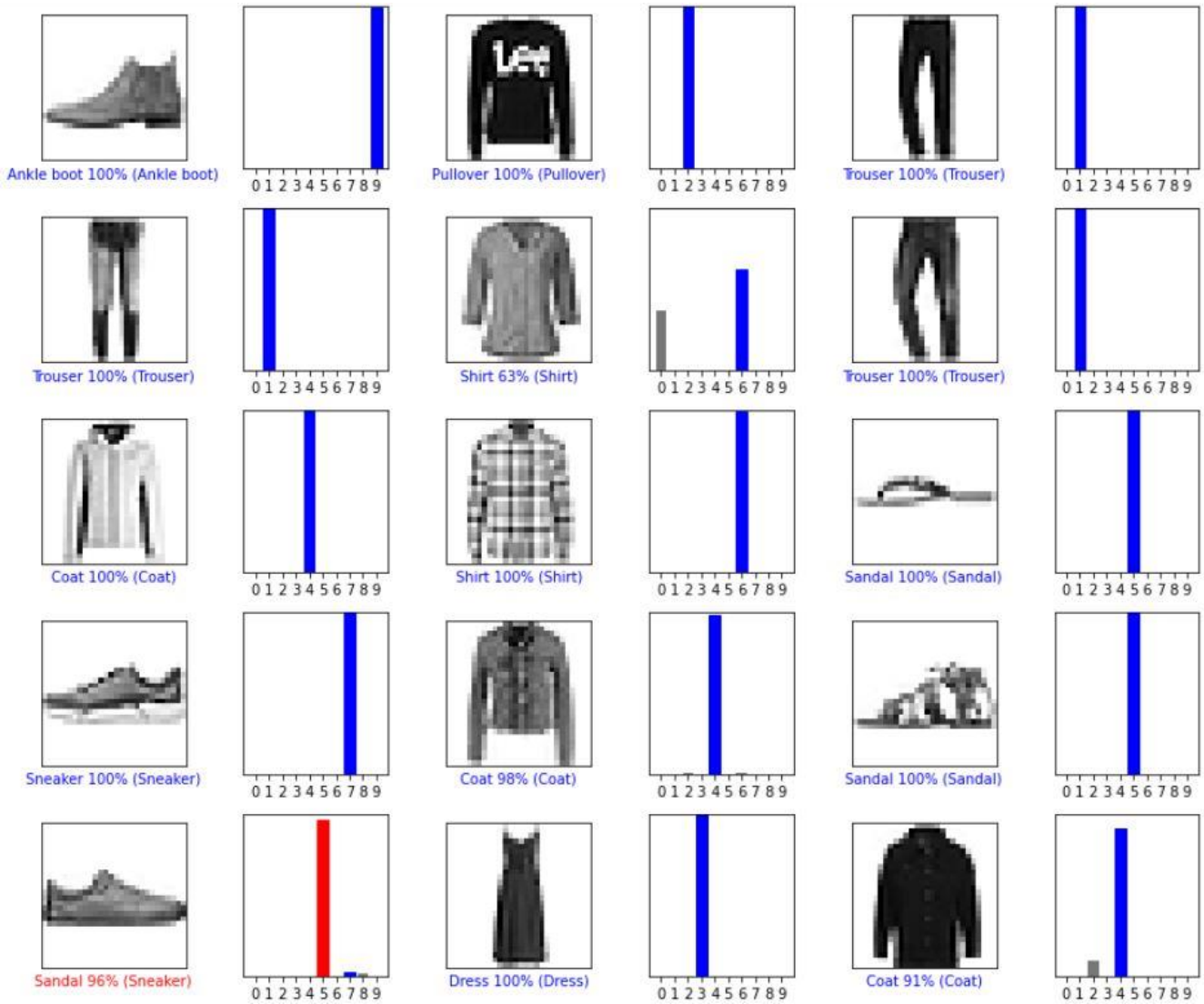
Nadam optimizer:



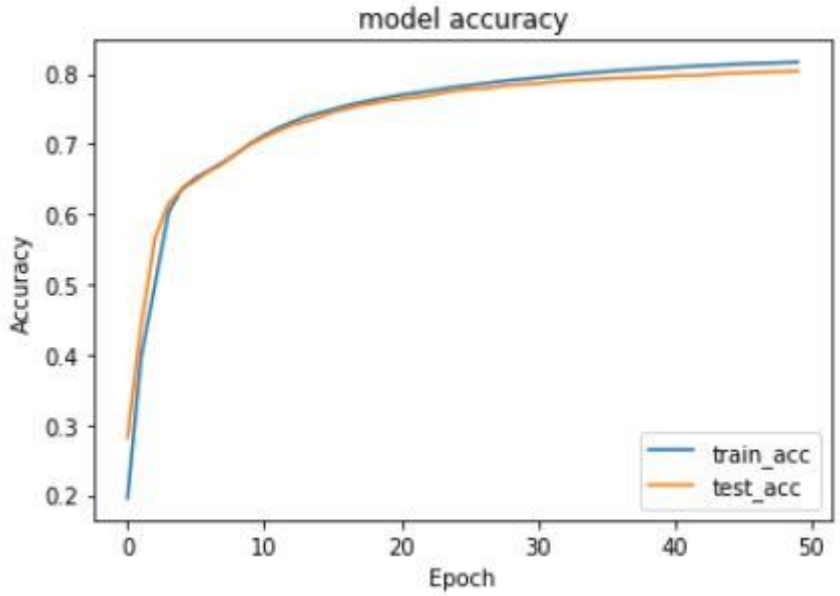


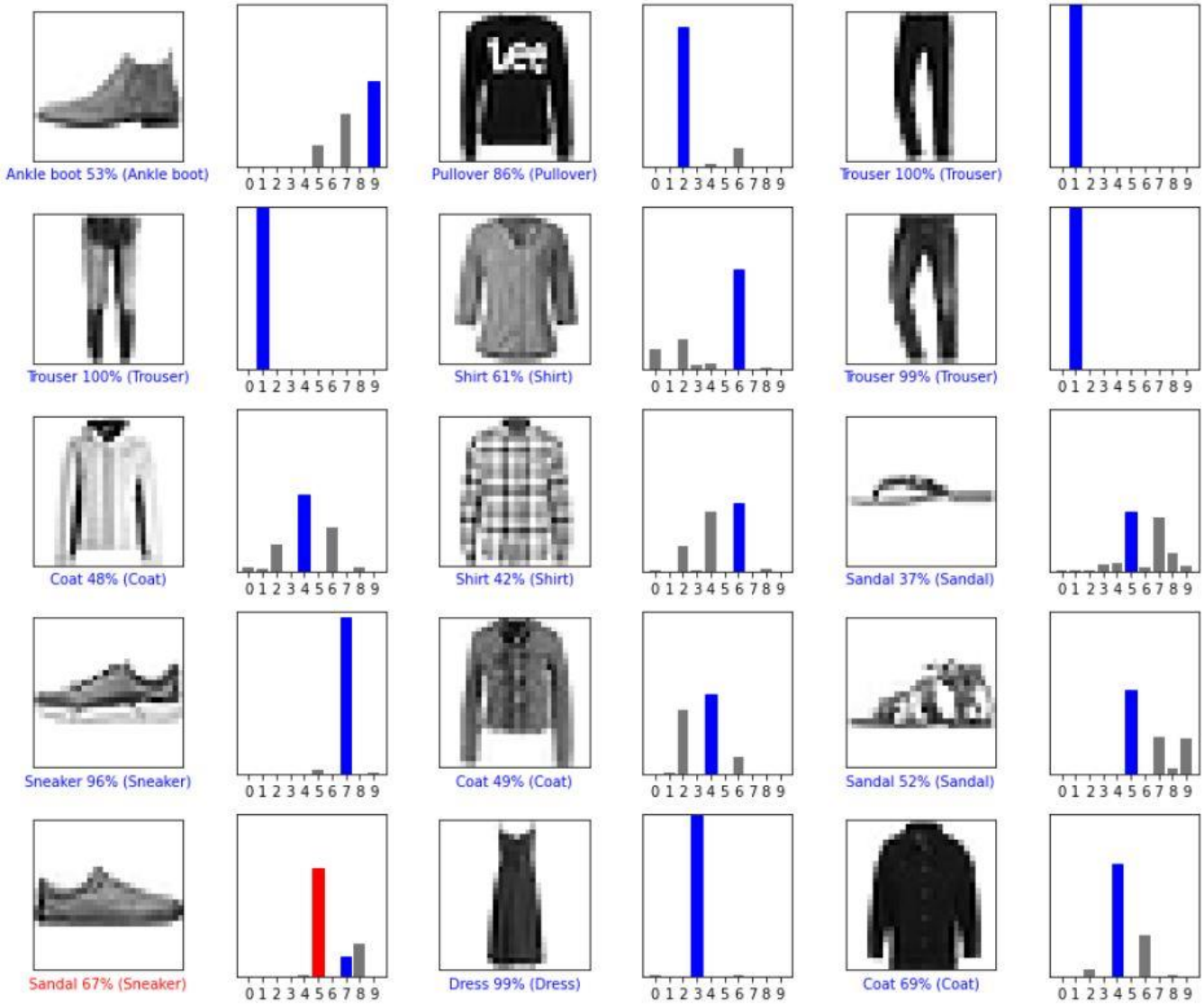
Adamax optimizer:





Ftrl optimizer:





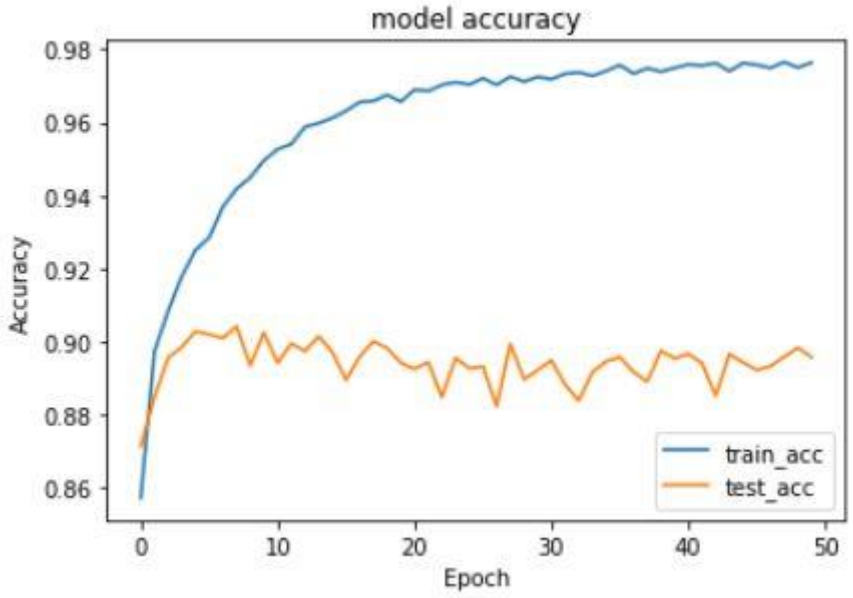
Παρακάτω φαίνονται οι τελικές ακρίβειες εκπαίδευσης και επικύρωσης των optimizers

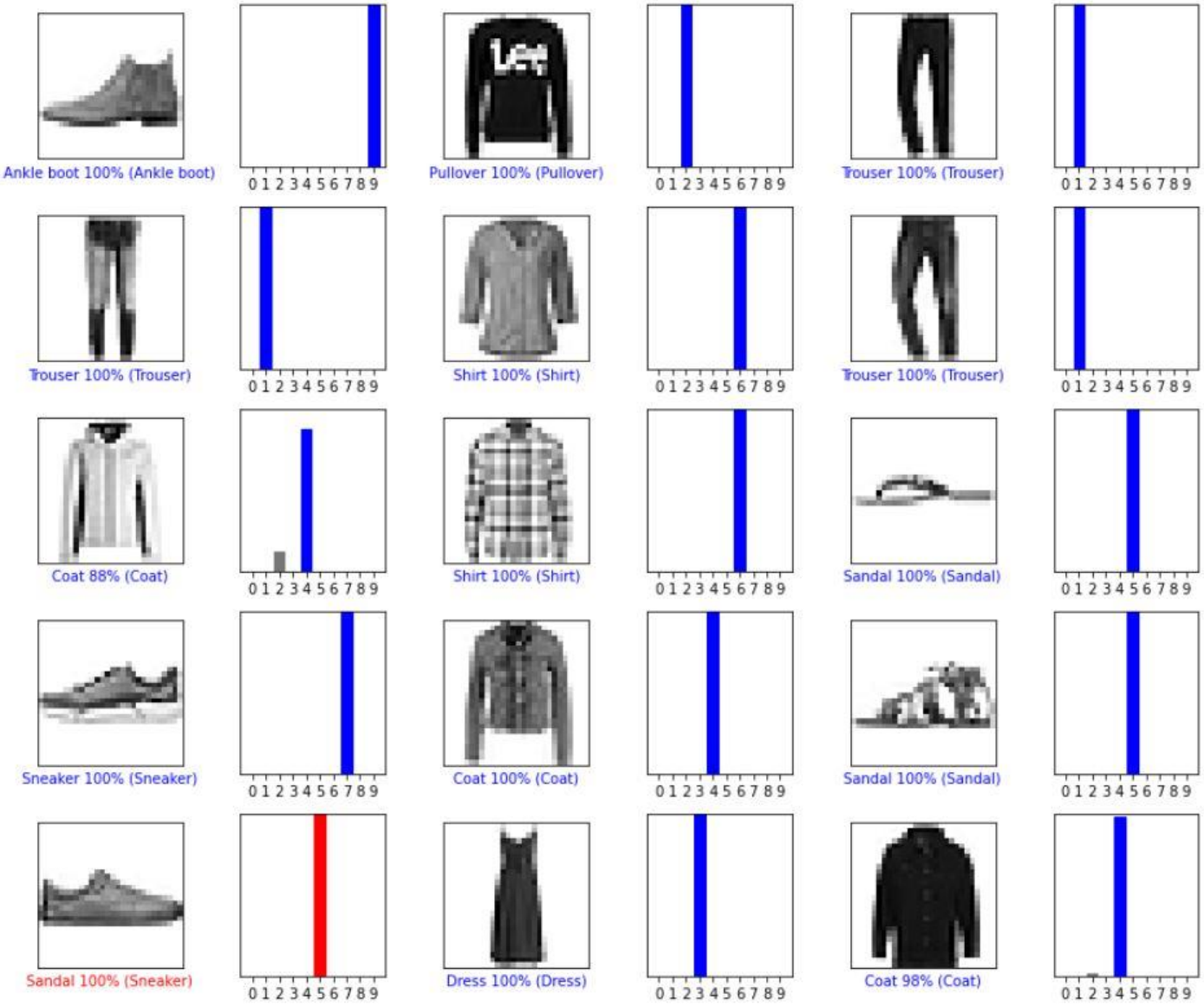
optimizer	train accuracy	validate accuracy
adam	96.44%	88.98%
sgd	91.43%	88.17%
rmsprop	94.59%	88.04%
nadam	96.56%	88.22%
adamax	94.76%	89.37%
ftrl	81.69%	80.42%

Παρατηρείται ότι ο καλύτερος αλγόριθμος βελτιστοποίησης είναι ο adam, καθώς πετυχαίνει το μεγαλύτερο άθροισμα train και validate accuracy, παρόλο που δεν πετυχαίνει συγκεκριμένα το καλύτερο train accuracy είτε το καλύτερο validate accuracy.

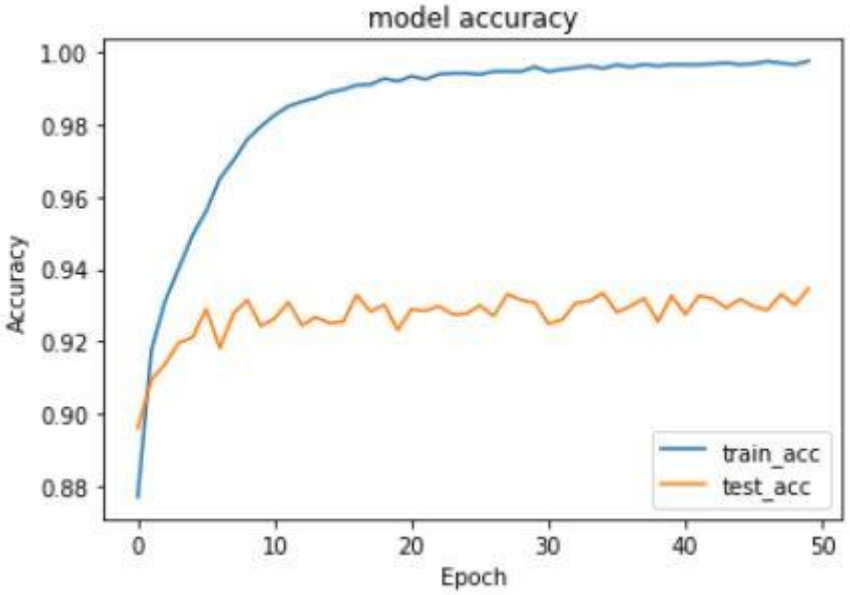
Παρακάτω αλλάζει η αρχιτεκτονική του δικτύου και κατασκευάζεται ένα καινούργιο στο αρχείο fashion_cnn, όπως φαίνεται στο σχήμα της εκφώνησης.

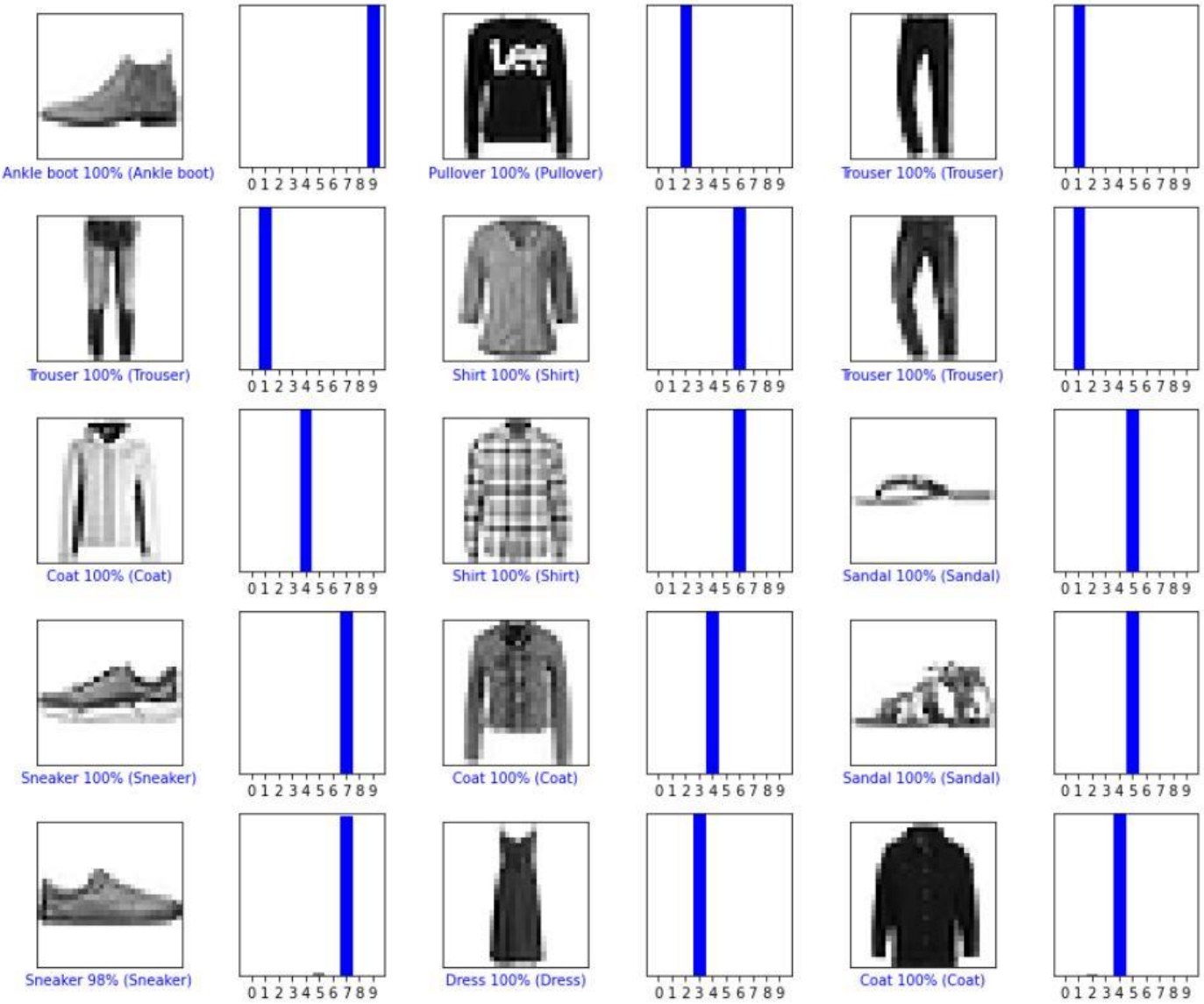
Αρχικά υλοποιήθηκε το νευρωνικό πριν μπει το batch normalization και παρακάτω παρατίθενται τα γραφήματα με τις τιμές ακρίβειας εκπαίδευσης καθώς και με τις τιμές ακρίβειας επικύρωσης, αλλά και οι προβλέψεις αναγνώρισης των προτύπων του μοντέλου:


model accuracy
Accuracy
Epoch
train_acc
test_acc
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0
10
20
30
40
50

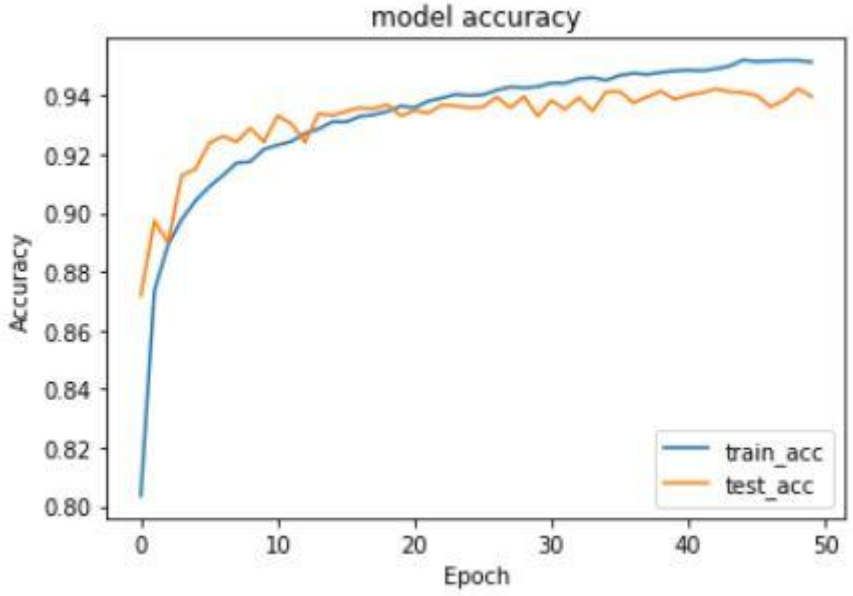

Ankle boot 100% (Ankle boot)
Pullover 100% (Pullover)
Trouser 100% (Trouser)
Trouser 100% (Trouser)
Shirt 100% (Shirt)
Trouser 100% (Trouser)
Coat 88% (Coat)
Shirt 100% (Shirt)
Sandal 100% (Sandal)
Sneaker 100% (Sneaker)
Coat 100% (Coat)
Sandal 100% (Sandal)
Sandal 100% (Sneaker)
Dress 100% (Dress)
Coat 98% (Coat)
0 1 2 3 4 5 6 7 8 9

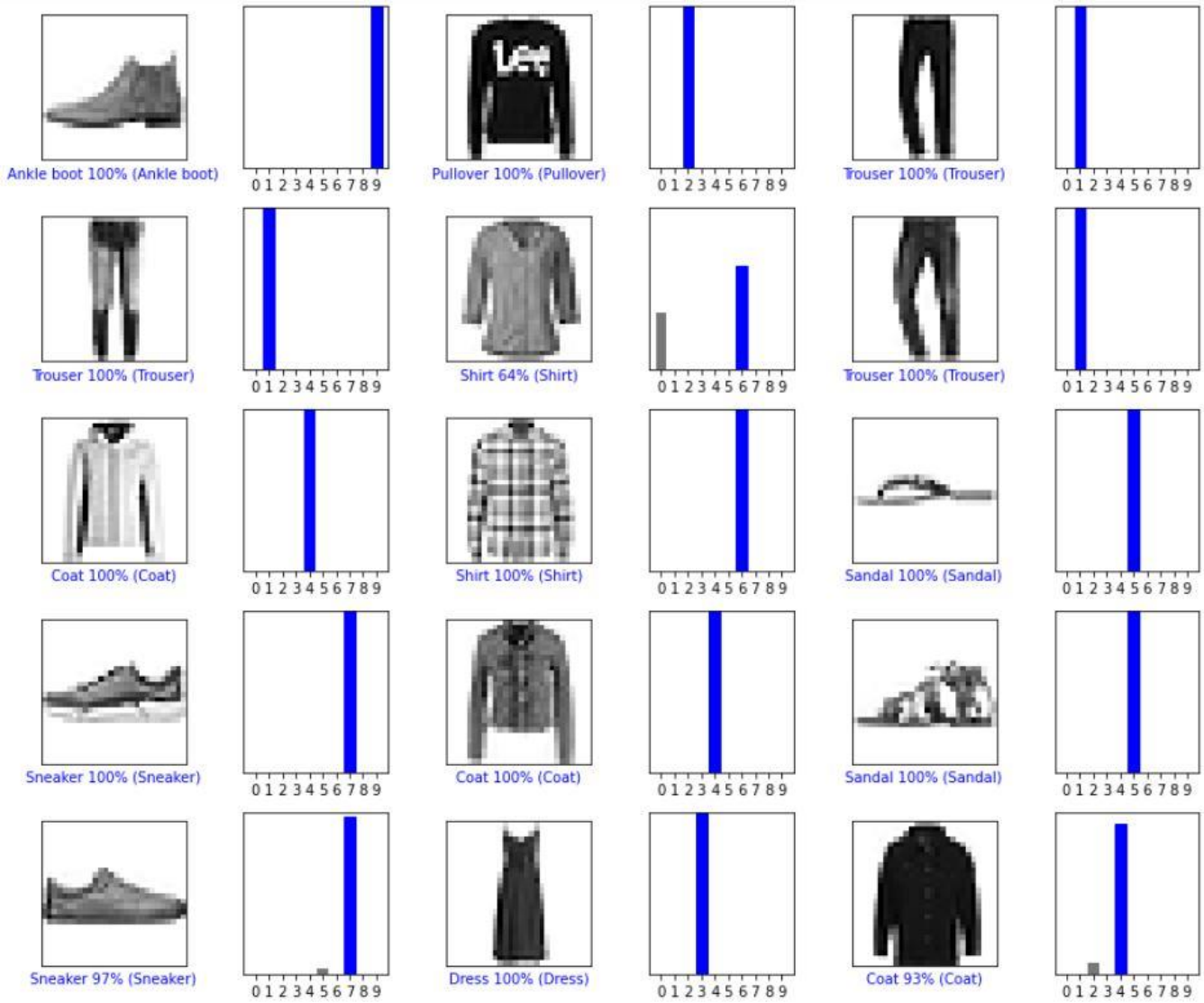
Υστερα προστίθεται και το batch normalization:





Τέλος προστίθεται και το dropout οπότε προκύπτει το τελικό νευρωνικό:





Παρατηρείται ότι όσο προστίθενται επίπεδα στο νευρωνικό, αυξάνεται η ακρίβεια επικύρωσης, ενώ τείνει να ταυτιστεί με την ακρίβεια εκπαίδευσης. Επίσης η προσθήκη του dropout μειώνει την ακρίβεια εκπαίδευσης, άρα μειώνεται και η τάση για overfitting.