Υπολογιστική Όραση και Γραφικά

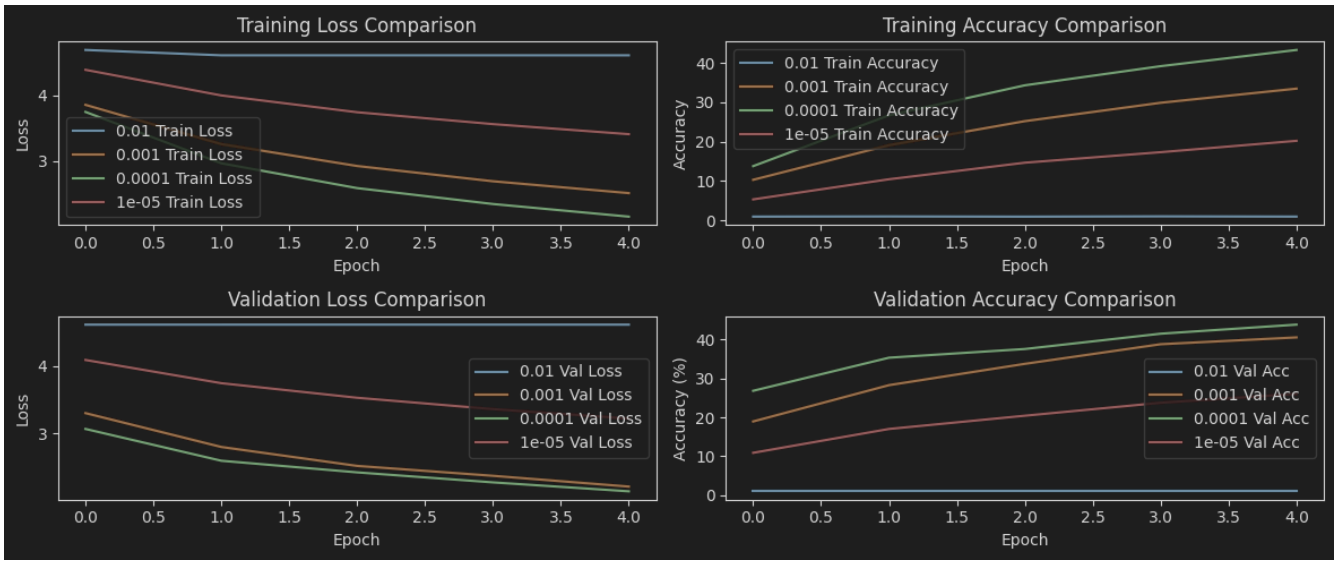
Ονοματεπώνυμο: Φώτης Χατζηφωτιάδης
ΑΜ: 2022116

Άσκηση #1, Βελτιστοποίηση αρχιτεκτονικής και υπερ-παραμέτρων:

Για διαφορετικά Learning Rate:

```
Training with LR=0.01

Epoch 1/5 | TrainLoss: 4.6903 | ValLoss: 4.6070 | TrainAcc: 0.93% | ValAcc: 1.00%

Epoch 2/5 | TrainLoss: 4.6089 | ValLoss: 4.6071 | TrainAcc: 0.97% | ValAcc: 1.00%

Epoch 3/5 | TrainLoss: 4.6091 | ValLoss: 4.6075 | TrainAcc: 0.92% | ValAcc: 1.00%

Epoch 4/5 | TrainLoss: 4.6092 | ValLoss: 4.6072 | TrainAcc: 0.99% | ValAcc: 1.00%

Epoch 5/5 | TrainLoss: 4.6089 | ValLoss: 4.6073 | TrainAcc: 0.93% | ValAcc: 1.00%
Training with LR=0.001
```

```
Training with LR=0.001

Epoch 1/5 | TrainLoss: 3.8562 | ValLoss: 3.3021 | TrainAcc: 10.29% | ValAcc: 18.94%

Epoch 2/5 | TrainLoss: 3.2595 | ValLoss: 2.8028 | TrainAcc: 19.08% | ValAcc: 28.26%

Epoch 3/5 | TrainLoss: 2.9269 | ValLoss: 2.5234 | TrainAcc: 25.19% | ValAcc: 33.73%

Epoch 4/5 | TrainLoss: 2.6949 | ValLoss: 2.3773 | TrainAcc: 29.87% | ValAcc: 38.77%

Epoch 5/5 | TrainLoss: 2.5147 | ValLoss: 2.2184 | TrainAcc: 33.46% | ValAcc: 40.52%
Training with LR=0.0001

Epoch 1/5 | TrainLoss: 3.7486 | ValLoss: 3.0686 | TrainAcc: 13.76% | ValAcc: 26.81%

Epoch 2/5 | TrainLoss: 2.9680 | ValLoss: 2.5995 | TrainAcc: 26.68% | ValAcc: 35.31%

Epoch 3/5 | TrainLoss: 2.5909 | ValLoss: 2.4280 | TrainAcc: 34.31% | ValAcc: 37.54%

Epoch 4/5 | TrainLoss: 2.3492 | ValLoss: 2.2803 | TrainAcc: 39.18% | ValAcc: 41.47%

Epoch 5/5 | TrainLoss: 2.1572 | ValLoss: 2.1485 | TrainAcc: 43.30% | ValAcc: 43.80%
Training with LR=1e-05

Epoch 1/5 | TrainLoss: 4.3883 | ValLoss: 4.0861 | TrainAcc: 5.31% | ValAcc: 10.89%

Epoch 2/5 | TrainLoss: 3.9991 | ValLoss: 3.7436 | TrainAcc: 10.43% | ValAcc: 17.01%

Epoch 3/5 | TrainLoss: 3.7439 | ValLoss: 3.5286 | TrainAcc: 14.63% | ValAcc: 20.41%

Epoch 4/5 | TrainLoss: 3.5634 | ValLoss: 3.3609 | TrainAcc: 17.31% | ValAcc: 23.74%

Epoch 5/5 | TrainLoss: 3.4105 | ValLoss: 3.2219 | TrainAcc: 20.20% | ValAcc: 26.02%
```



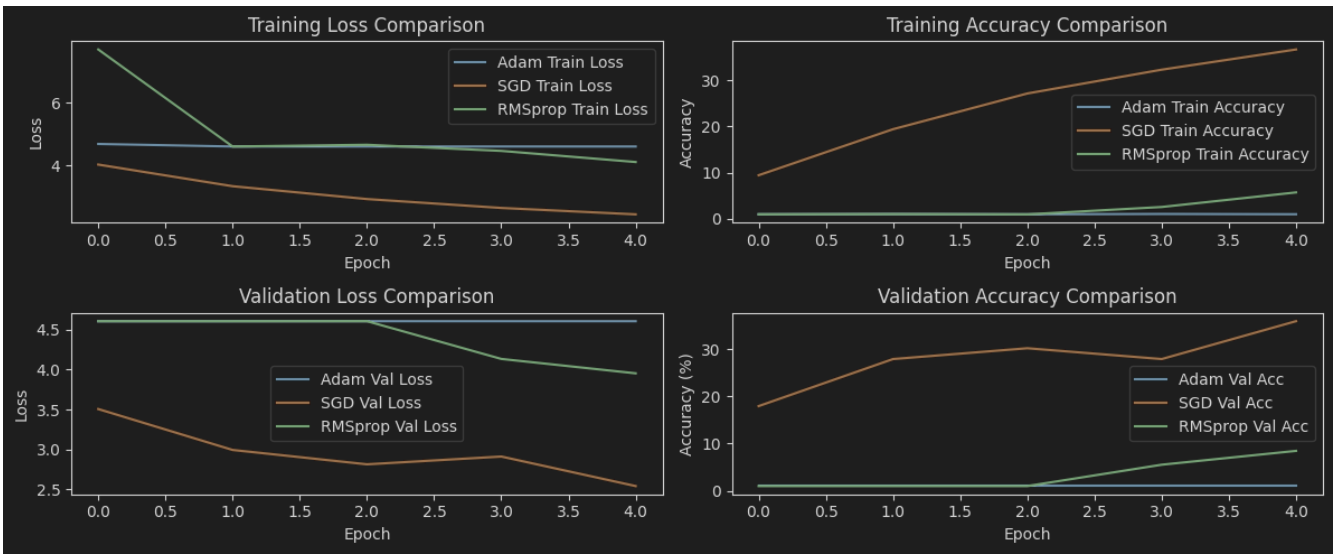
Σχόλιο:
Παρατηρώ ότι για learning rate = 1e-2 το δίκτυο δεν συγκλίνει και η ακρίβεια παραμένει περίπου στο 1%. Αυτό συμβαίνει επειδή η τιμή αυτή είναι υπερβολικά μεγάλη για τον Adam, προκαλώντας ασταθείς και υπερβολικά μεγάλες ενημερώσεις βαρών, με αποτέλεσμα την αποτυχία εκπαίδευσης.

Για τα learning rate = 1e-3,1e-4 τα αποτελέσματα είναι πιο σταθερά και ελεγχόμενα βήματα βελτιστοποίησης, επιτρέποντας στο δίκτυο να συγκλίνει ομαλά.

Για learning rate = 1e-5 η ακρίβεια ταξινόμησης είναι αρκετά μικρότερη, έχει αργότερη σύγκλιση στον ίδιο αριθμό εποχών εκπαίδευσης και εμφανίζει φαινόμενο υποεκπαίδευσης, κάτι που ευθύνεται στις μικρές ενημερώσεις των βαρών.

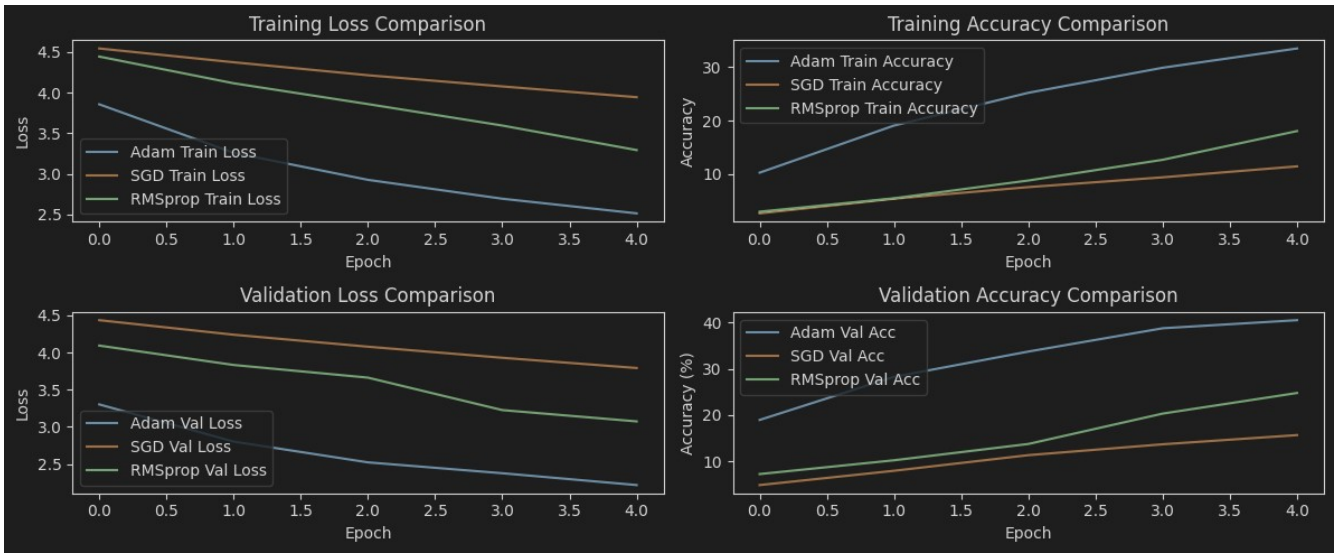
Για διαφορετικα optimizer:
Για learning rate=1e-2

```
Epoch 1/5 | TrainLoss: 4.6903 | ValLoss: 4.6070 | TrainAcc: 0.93% | ValAcc: 1.00%

Epoch 2/5 | TrainLoss: 4.6089 | ValLoss: 4.6071 | TrainAcc: 0.97% | ValAcc: 1.00%

Epoch 3/5 | TrainLoss: 4.6091 | ValLoss: 4.6075 | TrainAcc: 0.92% | ValAcc: 1.00%

Epoch 4/5 | TrainLoss: 4.6092 | ValLoss: 4.6072 | TrainAcc: 0.99% | ValAcc: 1.00%

Epoch 5/5 | TrainLoss: 4.6089 | ValLoss: 4.6073 | TrainAcc: 0.93% | ValAcc: 1.00%
Training with SGD

Epoch 1/5 | TrainLoss: 4.0334 | ValLoss: 3.5059 | TrainAcc: 9.42% | ValAcc: 17.93%

Epoch 2/5 | TrainLoss: 3.3448 | ValLoss: 2.9930 | TrainAcc: 19.42% | ValAcc: 27.89%

Epoch 3/5 | TrainLoss: 2.9360 | ValLoss: 2.8128 | TrainAcc: 27.19% | ValAcc: 30.18%

Epoch 4/5 | TrainLoss: 2.6528 | ValLoss: 2.9104 | TrainAcc: 32.37% | ValAcc: 27.89%

Epoch 5/5 | TrainLoss: 2.4486 | ValLoss: 2.5403 | TrainAcc: 36.76% | ValAcc: 35.91%
Training with RMSprop

Epoch 1/5 | TrainLoss: 7.6981 | ValLoss: 4.6075 | TrainAcc: 0.92% | ValAcc: 1.00%

Epoch 2/5 | TrainLoss: 4.6091 | ValLoss: 4.6072 | TrainAcc: 0.96% | ValAcc: 1.00%

Epoch 3/5 | TrainLoss: 4.6639 | ValLoss: 4.6077 | TrainAcc: 0.91% | ValAcc: 1.00%

Epoch 4/5 | TrainLoss: 4.4690 | ValLoss: 4.1342 | TrainAcc: 2.51% | ValAcc: 5.52%

Epoch 5/5 | TrainLoss: 4.1147 | ValLoss: 3.9538 | TrainAcc: 5.68% | ValAcc: 8.45%
```



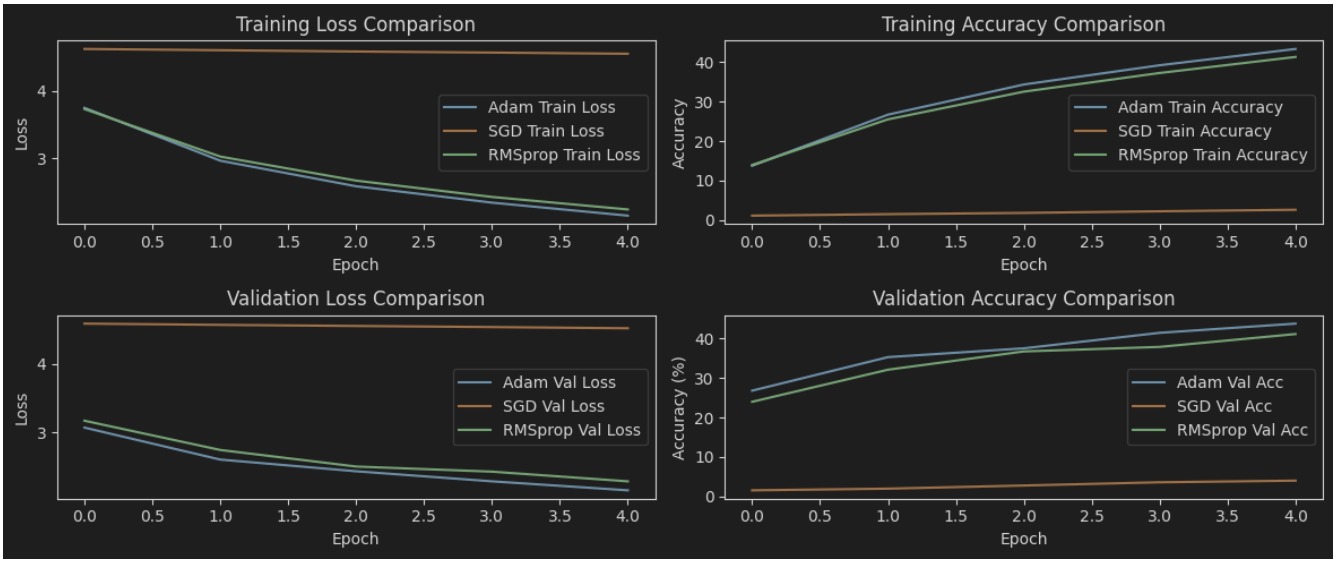
Mε learning rate = 1e-3:

```
Training with Adam

Epoch 1/5 | TrainLoss: 3.8562 | ValLoss: 3.3021 | TrainAcc: 10.29% | ValAcc: 18.94%

Epoch 2/5 | TrainLoss: 3.2595 | ValLoss: 2.8028 | TrainAcc: 19.08% | ValAcc: 28.26%

Epoch 3/5 | TrainLoss: 2.9269 | ValLoss: 2.5234 | TrainAcc: 25.19% | ValAcc: 33.73%

Epoch 4/5 | TrainLoss: 2.6949 | ValLoss: 2.3773 | TrainAcc: 29.87% | ValAcc: 38.77%

Epoch 5/5 | TrainLoss: 2.5147 | ValLoss: 2.2184 | TrainAcc: 33.46% | ValAcc: 40.52%
Training with SGD

Epoch 1/5 | TrainLoss: 4.5429 | ValLoss: 4.4343 | TrainAcc: 2.69% | ValAcc: 4.85%

Epoch 2/5 | TrainLoss: 4.3732 | ValLoss: 4.2389 | TrainAcc: 5.43% | ValAcc: 7.94%

Epoch 3/5 | TrainLoss: 4.2136 | ValLoss: 4.0778 | TrainAcc: 7.59% | ValAcc: 11.32%

Epoch 4/5 | TrainLoss: 4.0762 | ValLoss: 3.9288 | TrainAcc: 9.42% | ValAcc: 13.67%

Epoch 5/5 | TrainLoss: 3.9437 | ValLoss: 3.7914 | TrainAcc: 11.47% | ValAcc: 15.66%
Training with RMSprop

Epoch 1/5 | TrainLoss: 4.4432 | ValLoss: 4.0926 | TrainAcc: 3.01% | ValAcc: 7.23%

Epoch 2/5 | TrainLoss: 4.1149 | ValLoss: 3.8321 | TrainAcc: 5.50% | ValAcc: 10.21%

Epoch 3/5 | TrainLoss: 3.8587 | ValLoss: 3.6619 | TrainAcc: 8.82% | ValAcc: 13.74%

Epoch 4/5 | TrainLoss: 3.5949 | ValLoss: 3.2259 | TrainAcc: 12.69% | ValAcc: 20.31%

Epoch 5/5 | TrainLoss: 3.2931 | ValLoss: 3.0738 | TrainAcc: 18.05% | ValAcc: 24.77%
```



Για learning rate 1e-4:

```
Training with Adam

Epoch 1/5 | TrainLoss: 3.7486 | ValLoss: 3.0686 | TrainAcc: 13.76% | ValAcc: 26.81%

Epoch 2/5 | TrainLoss: 2.9680 | ValLoss: 2.5995 | TrainAcc: 26.68% | ValAcc: 35.31%

Epoch 3/5 | TrainLoss: 2.5909 | ValLoss: 2.4280 | TrainAcc: 34.31% | ValAcc: 37.54%

Epoch 4/5 | TrainLoss: 2.3492 | ValLoss: 2.2803 | TrainAcc: 39.18% | ValAcc: 41.47%

Epoch 5/5 | TrainLoss: 2.1572 | ValLoss: 2.1485 | TrainAcc: 43.30% | ValAcc: 43.80%
Training with SGD

Epoch 1/5 | TrainLoss: 4.6179 | ValLoss: 4.5941 | TrainAcc: 1.13% | ValAcc: 1.55%

Epoch 2/5 | TrainLoss: 4.5991 | ValLoss: 4.5760 | TrainAcc: 1.51% | ValAcc: 1.97%

Epoch 3/5 | TrainLoss: 4.5802 | ValLoss: 4.5597 | TrainAcc: 1.83% | ValAcc: 2.77%

Epoch 4/5 | TrainLoss: 4.5643 | ValLoss: 4.5437 | TrainAcc: 2.24% | ValAcc: 3.59%

Epoch 5/5 | TrainLoss: 4.5469 | ValLoss: 4.5261 | TrainAcc: 2.62% | ValAcc: 4.00%
Training with RMSprop

Epoch 1/5 | TrainLoss: 3.7305 | ValLoss: 3.1696 | TrainAcc: 13.96% | ValAcc: 23.99%

Epoch 2/5 | TrainLoss: 3.0293 | ValLoss: 2.7404 | TrainAcc: 25.49% | ValAcc: 32.12%

Epoch 3/5 | TrainLoss: 2.6746 | ValLoss: 2.4974 | TrainAcc: 32.50% | ValAcc: 36.75%

Epoch 4/5 | TrainLoss: 2.4332 | ValLoss: 2.4224 | TrainAcc: 37.22% | ValAcc: 37.90%

Epoch 5/5 | TrainLoss: 2.2486 | ValLoss: 2.2803 | TrainAcc: 41.28% | ValAcc: 41.16%
```

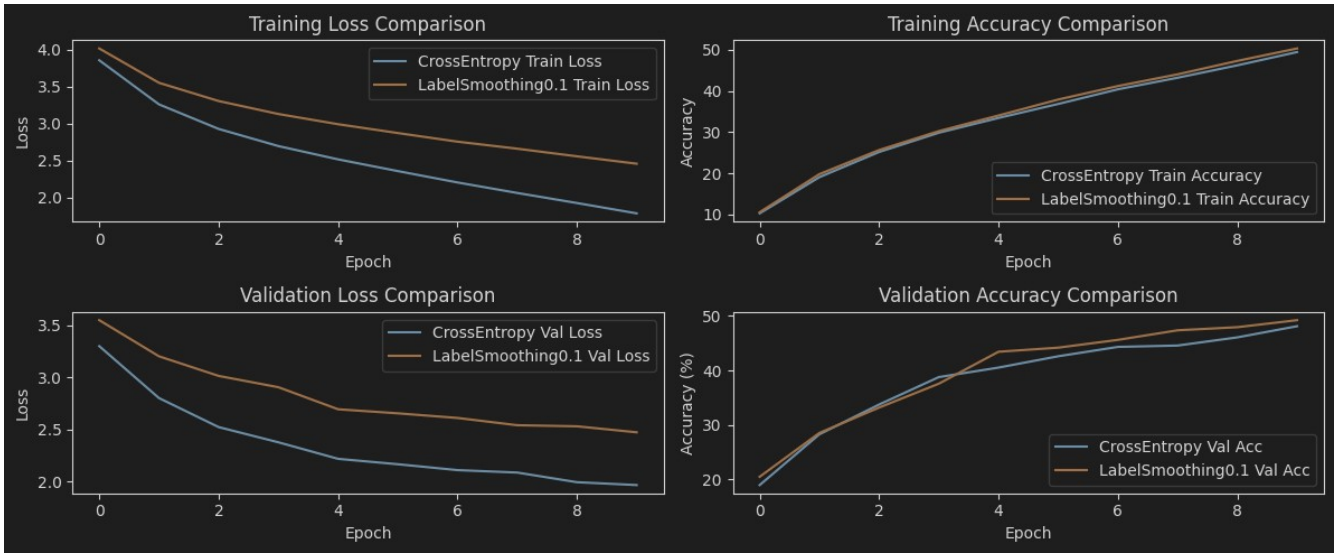


Σχόλιο:
Αυτό που παρατηρείται είναι ότι δεν μπορεί να γίνει άμεση σύγκριση των optimizer καθώς όπως φαίνεται στο παραπάνω παράδειγμα ο κάθε optimizer έχει διαφορετικά αποτελέσματα με διαφορετικά learning rate. Ο λόγος που δοκίμασα και για τα 3 optimizer διαφορετικά learning rates είναι επειδή μου φαινόταν υπερβολικά μικρό το accuracy του SGD για lr=1e-3 που δοκίμαζα στην αρχή. Είδα αυτή τη συζήτηση και είπα να δοκιμάσω για διαφορετικές τιμές.

Διαφορές CrossEntropy με και χωρίς label smoothing:

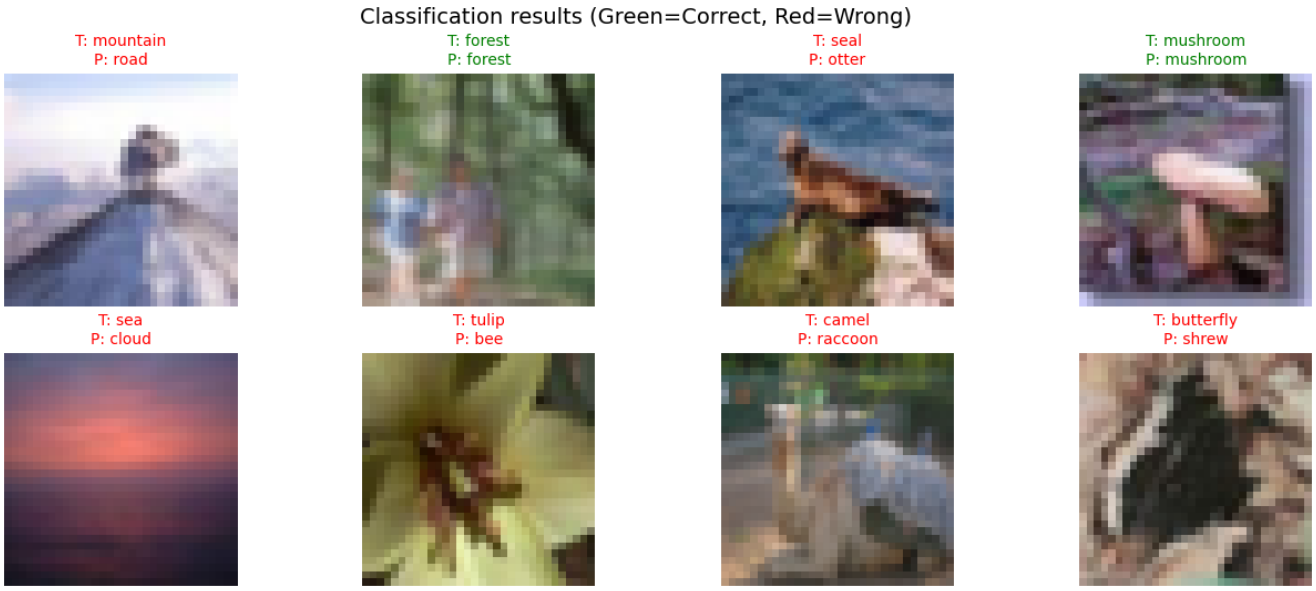
```
Training with CrossEntropy

Epoch 1/10 | TrainLoss: 3.8562 | ValLoss: 3.3021 | TrainAcc: 10.29% | ValAcc: 18.94%

Epoch 2/10 | TrainLoss: 3.2595 | ValLoss: 2.8028 | TrainAcc: 19.08% | ValAcc: 28.26%

Epoch 3/10 | TrainLoss: 2.9269 | ValLoss: 2.5234 | TrainAcc: 25.19% | ValAcc: 33.73%

Epoch 4/10 | TrainLoss: 2.6949 | ValLoss: 2.3773 | TrainAcc: 29.87% | ValAcc: 38.77%

Epoch 5/10 | TrainLoss: 2.5147 | ValLoss: 2.2184 | TrainAcc: 33.46% | ValAcc: 40.52%

Epoch 6/10 | TrainLoss: 2.3563 | ValLoss: 2.1665 | TrainAcc: 36.87% | ValAcc: 42.60%

Epoch 7/10 | TrainLoss: 2.2023 | ValLoss: 2.1099 | TrainAcc: 40.42% | ValAcc: 44.32%

Epoch 8/10 | TrainLoss: 2.0603 | ValLoss: 2.0865 | TrainAcc: 43.24% | ValAcc: 44.57%

Epoch 9/10 | TrainLoss: 1.9240 | ValLoss: 1.9937 | TrainAcc: 46.25% | ValAcc: 46.07%

Epoch 10/10 | TrainLoss: 1.7842 | ValLoss: 1.9662 | TrainAcc: 49.47% | ValAcc: 48.12%
```

```
Training with LabelSmoothing0.1

Epoch 1/10 | TrainLoss: 4.0161 | ValLoss: 3.5518 | TrainAcc: 10.55% | ValAcc: 20.46%

Epoch 2/10 | TrainLoss: 3.5513 | ValLoss: 3.2044 | TrainAcc: 19.82% | ValAcc: 28.51%

Epoch 3/10 | TrainLoss: 3.3045 | ValLoss: 3.0156 | TrainAcc: 25.67% | ValAcc: 33.17%

Epoch 4/10 | TrainLoss: 3.1290 | ValLoss: 2.9070 | TrainAcc: 30.26% | ValAcc: 37.56%

Epoch 5/10 | TrainLoss: 2.9908 | ValLoss: 2.6950 | TrainAcc: 34.07% | ValAcc: 43.43%

Epoch 6/10 | TrainLoss: 2.8713 | ValLoss: 2.6560 | TrainAcc: 37.94% | ValAcc: 44.18%

Epoch 7/10 | TrainLoss: 2.7546 | ValLoss: 2.6114 | TrainAcc: 41.22% | ValAcc: 45.61%

Epoch 8/10 | TrainLoss: 2.6595 | ValLoss: 2.5416 | TrainAcc: 44.07% | ValAcc: 47.37%

Epoch 9/10 | TrainLoss: 2.5563 | ValLoss: 2.5312 | TrainAcc: 47.33% | ValAcc: 47.94%

Epoch 10/10 | TrainLoss: 2.4561 | ValLoss: 2.4734 | TrainAcc: 50.35% | ValAcc: 49.23%
```



Σχόλιο:
Σε αυτο το παράδειγμα έτρεξα 10 epochs αντι για 5 που ετρεχα στα προηγούμενα γιατί απο το 4ο epoch και μετά το Cross Entropy με το LabelSmoothing=0.1 αρχίζει και έχει καλύτερο accuracy και ήθελα να δω άμα είναι είναι μια τάση που συνεχίζεται ή μια στατιστική απόκλιση για 2 epochs(4ο και 5ο). Το αποτέλεσμα είναι ότι το CrossEntropy με LS παρόλο που έχει υψηλότερο loss, δεν λειτουργεί περιοριστικά και ίσα ίσα λόγω της γενίκευσης του μοντέλου, μειώνει την “αυτοπεποίθηση” του μοντέλου και καταλήγει σε σωστότερα αποτελέσματα.

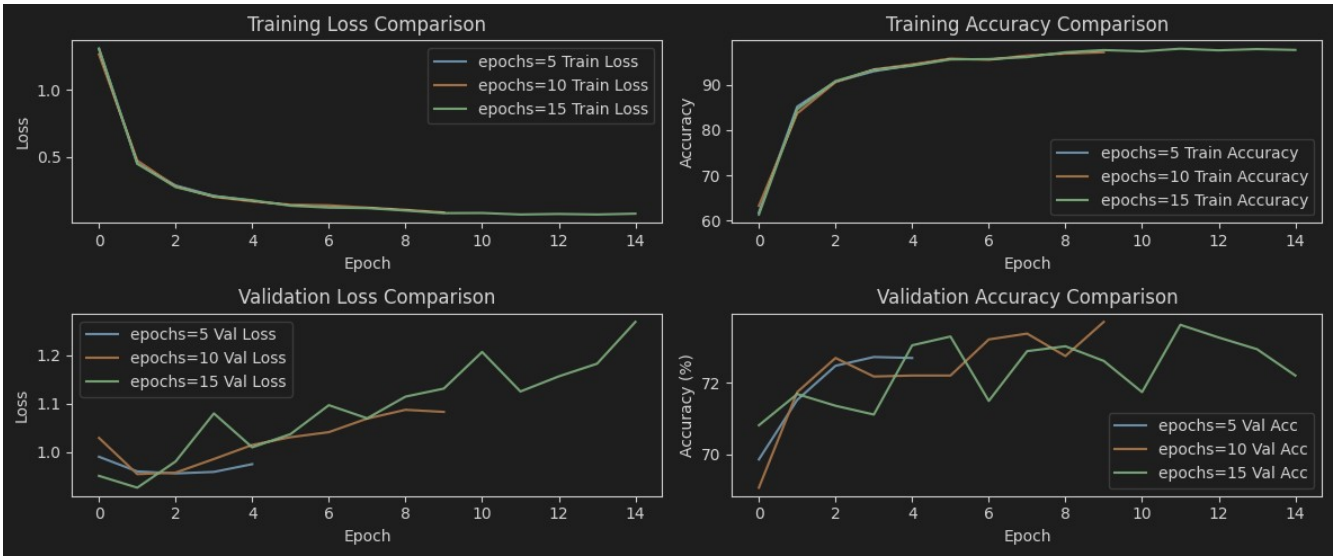
Ενδεικτικά αποτελέσματα (υπολογισμένα με optimizer = Adam, loss=CrossEntropyLoss(label_smoothing=0.1, learning_rate=1e-3, epochs=10)



Άσκηση #2
Για διαφορετικά epochs:

```
Training with epochs=5

Epoch 1/5 | TrainLoss: 1.3114 | ValLoss: 0.9902 | TrainAcc: 61.82% | ValAcc: 69.86%

Epoch 2/5 | TrainLoss: 0.4483 | ValLoss: 0.9598 | TrainAcc: 85.19% | ValAcc: 71.52%

Epoch 3/5 | TrainLoss: 0.2842 | ValLoss: 0.9557 | TrainAcc: 90.65% | ValAcc: 72.47%

Epoch 4/5 | TrainLoss: 0.2048 | ValLoss: 0.9589 | TrainAcc: 92.96% | ValAcc: 72.72%

Epoch 5/5 | TrainLoss: 0.1689 | ValLoss: 0.9749 | TrainAcc: 94.35% | ValAcc: 72.69%

Training with epochs=10

Epoch 1/10 | TrainLoss: 1.2689 | ValLoss: 1.0293 | TrainAcc: 63.32% | ValAcc: 69.07%

Epoch 2/10 | TrainLoss: 0.4708 | ValLoss: 0.9542 | TrainAcc: 83.67% | ValAcc: 71.74%

Epoch 3/10 | TrainLoss: 0.2776 | ValLoss: 0.9574 | TrainAcc: 90.57% | ValAcc: 72.69%

Epoch 4/10 | TrainLoss: 0.1978 | ValLoss: 0.9855 | TrainAcc: 93.37% | ValAcc: 72.17%

Epoch 5/10 | TrainLoss: 0.1644 | ValLoss: 1.0147 | TrainAcc: 94.48% | ValAcc: 72.20%

Epoch 6/10 | TrainLoss: 0.1389 | ValLoss: 1.0307 | TrainAcc: 95.76% | ValAcc: 72.20%

Epoch 7/10 | TrainLoss: 0.1345 | ValLoss: 1.0411 | TrainAcc: 95.49% | ValAcc: 73.21%

Epoch 8/10 | TrainLoss: 0.1173 | ValLoss: 1.0692 | TrainAcc: 96.44% | ValAcc: 73.37%

Epoch 9/10 | TrainLoss: 0.1009 | ValLoss: 1.0874 | TrainAcc: 96.88% | ValAcc: 72.74%

Epoch 10/10 | TrainLoss: 0.0808 | ValLoss: 1.0830 | TrainAcc: 97.15% | ValAcc: 73.70%
```

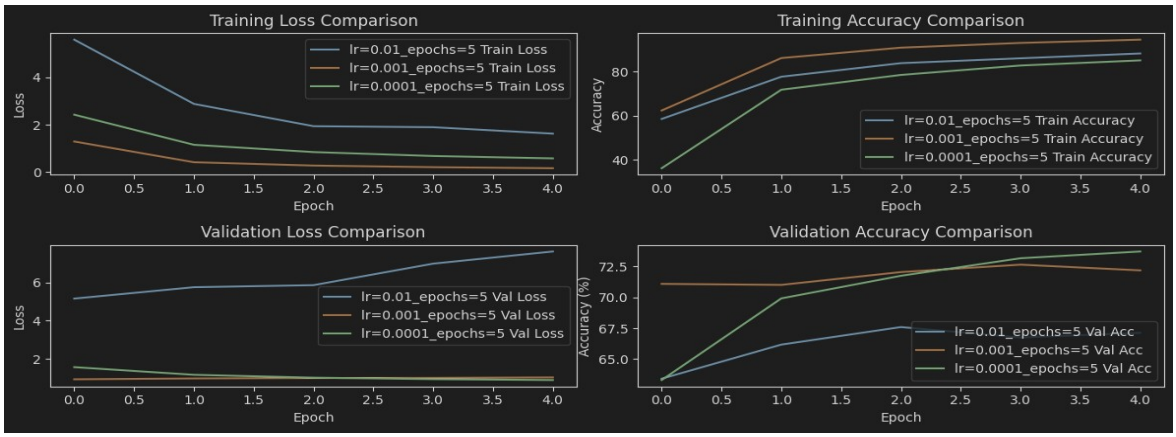
```
Training with epochs=15

Epoch 1/15 | TrainLoss: 1.3115 | ValLoss: 0.9507 | TrainAcc: 61.39% | ValAcc: 70.81%

Epoch 2/15 | TrainLoss: 0.4473 | ValLoss: 0.9262 | TrainAcc: 84.59% | ValAcc: 71.68%

Epoch 3/15 | TrainLoss: 0.2718 | ValLoss: 0.9804 | TrainAcc: 90.82% | ValAcc: 71.35%

Epoch 4/15 | TrainLoss: 0.2030 | ValLoss: 1.0799 | TrainAcc: 93.23% | ValAcc: 71.11%

Epoch 5/15 | TrainLoss: 0.1725 | ValLoss: 1.0097 | TrainAcc: 94.18% | ValAcc: 73.04%

Epoch 6/15 | TrainLoss: 0.1316 | ValLoss: 1.0372 | TrainAcc: 95.57% | ValAcc: 73.29%

Epoch 7/15 | TrainLoss: 0.1172 | ValLoss: 1.0970 | TrainAcc: 95.65% | ValAcc: 71.49%

Epoch 8/15 | TrainLoss: 0.1136 | ValLoss: 1.0698 | TrainAcc: 96.09% | ValAcc: 72.88%

Epoch 9/15 | TrainLoss: 0.0953 | ValLoss: 1.1147 | TrainAcc: 97.15% | ValAcc: 73.02%

Epoch 10/15 | TrainLoss: 0.0758 | ValLoss: 1.1313 | TrainAcc: 97.64% | ValAcc: 72.61%

Epoch 11/15 | TrainLoss: 0.0769 | ValLoss: 1.2071 | TrainAcc: 97.36% | ValAcc: 71.74%

Epoch 12/15 | TrainLoss: 0.0656 | ValLoss: 1.1253 | TrainAcc: 97.93% | ValAcc: 73.62%

Epoch 13/15 | TrainLoss: 0.0697 | ValLoss: 1.1566 | TrainAcc: 97.58% | ValAcc: 73.26%

Epoch 14/15 | TrainLoss: 0.0655 | ValLoss: 1.1829 | TrainAcc: 97.85% | ValAcc: 72.94%

Epoch 15/15 | TrainLoss: 0.0720 | ValLoss: 1.2694 | TrainAcc: 97.66% | ValAcc: 72.20%
```



Σχόλιο:
Το μοντέλο μαθαίνει γρήγορα καθώς το train accuracy είναι υψηλό ήδη από τα πρώτα epochs, 61.82% → 94.35% . Το validation accuracy δεν ακολουθεί την ίδια τάση και παραμένει κοντά στα ποσοστά με το πρώτο epoch, απο 69.86% → 72.69%. Το μοντέλο δεν εκμεταλλεύεται πλήρως την χωρητικότητα του δικτύου και οδηγεί σε overfitting.

Για διαφορετικά learning rates:

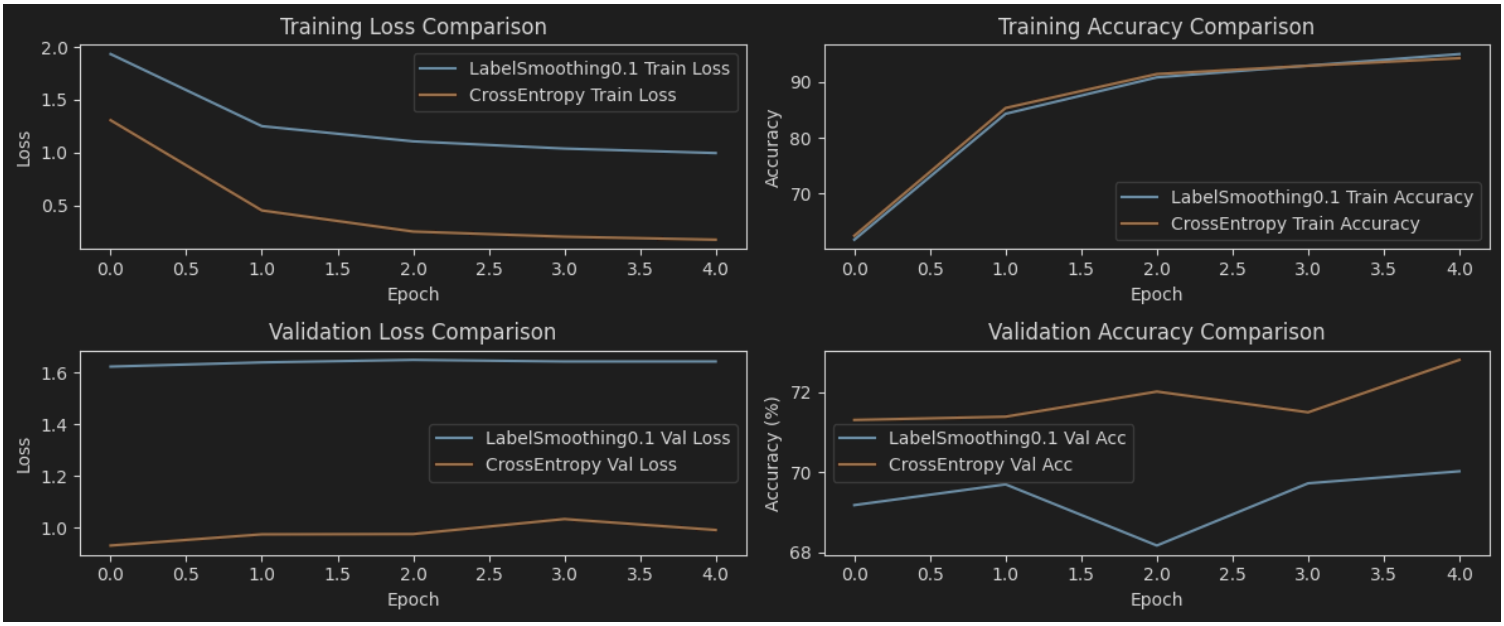
```
TRAINING WITH LR=0.01

Epoch 1/5 | TrainLoss: 5.5966 | ValLoss: 5.1509 | TrainAcc: 58.48% | ValAcc: 63.42%

Epoch 2/5 | TrainLoss: 2.8828 | ValLoss: 5.7478 | TrainAcc: 77.58% | ValAcc: 66.18%

Epoch 3/5 | TrainLoss: 1.9404 | ValLoss: 5.8550 | TrainAcc: 83.72% | ValAcc: 67.59%

Epoch 4/5 | TrainLoss: 1.8979 | ValLoss: 6.9805 | TrainAcc: 85.92% | ValAcc: 66.75%

Epoch 5/5 | TrainLoss: 1.6260 | ValLoss: 7.6173 | TrainAcc: 88.10% | ValAcc: 67.13%
TRAINING WITH LR=0.001

Epoch 1/5 | TrainLoss: 1.2933 | ValLoss: 0.9239 | TrainAcc: 62.31% | ValAcc: 71.08%

Epoch 2/5 | TrainLoss: 0.4175 | ValLoss: 0.9687 | TrainAcc: 86.03% | ValAcc: 71.00%

Epoch 3/5 | TrainLoss: 0.2762 | ValLoss: 0.9908 | TrainAcc: 90.73% | ValAcc: 72.04%

Epoch 4/5 | TrainLoss: 0.2087 | ValLoss: 0.9985 | TrainAcc: 92.88% | ValAcc: 72.64%

Epoch 5/5 | TrainLoss: 0.1647 | ValLoss: 1.0236 | TrainAcc: 94.35% | ValAcc: 72.17%
TRAINING WITH LR=0.0001

Epoch 1/5 | TrainLoss: 2.4259 | ValLoss: 1.5610 | TrainAcc: 36.20% | ValAcc: 63.31%

Epoch 2/5 | TrainLoss: 1.1541 | ValLoss: 1.1601 | TrainAcc: 71.71% | ValAcc: 69.91%

Epoch 3/5 | TrainLoss: 0.8464 | ValLoss: 1.0105 | TrainAcc: 78.40% | ValAcc: 71.74%

Epoch 4/5 | TrainLoss: 0.6811 | ValLoss: 0.9309 | TrainAcc: 82.66% | ValAcc: 73.15%

Epoch 5/5 | TrainLoss: 0.5809 | ValLoss: 0.8859 | TrainAcc: 84.97% | ValAcc: 73.70%
```



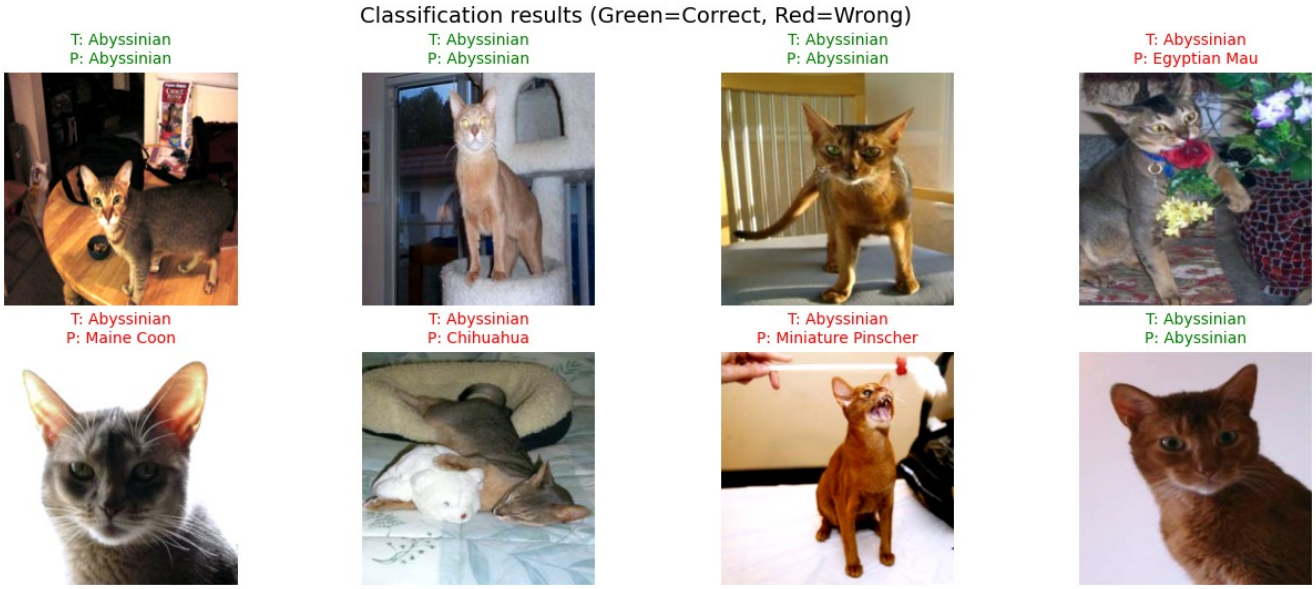
Σχόλιο:
Η εκπαίδευση γίνεται πάλι με optimizer τον Adam και τα αποτελέσματα είναι παρόμοια καθώς και επαναλαμβάνεται το μοτίβο 1e-4>1e-3>1e-2. Ο λόγος που δεν είναι το acc του 1e-2 κοντά στο 1% όπως στο προηγούμενο παράδειγμα είναι ότι τώρα παρεμβαίνει μόνο στο τελευταίο layer, `optimizer = torch.optim.Adam(model.classifier[6].parameters(), lr=lr)` και λόγω του προεκπαιδευμένου backbone συνεχίζει να εξάγει χρήσιμα χαρακτηριστικά.

Με και χωρίς label smoothing:

```
Training with LabelSmoothing0.1

Epoch 1/5 | TrainLoss: 1.9322 | ValLoss: 1.6242 | TrainAcc: 61.82% | ValAcc: 69.17%

Epoch 2/5 | TrainLoss: 1.2508 | ValLoss: 1.6405 | TrainAcc: 84.27% | ValAcc: 69.69%

Epoch 3/5 | TrainLoss: 1.1074 | ValLoss: 1.6502 | TrainAcc: 90.76% | ValAcc: 68.17%

Epoch 4/5 | TrainLoss: 1.0390 | ValLoss: 1.6440 | TrainAcc: 92.85% | ValAcc: 69.72%

Epoch 5/5 | TrainLoss: 0.9967 | ValLoss: 1.6441 | TrainAcc: 94.92% | ValAcc: 70.02%

--- Αποτελέσματα Loss Functions ---
Training with CrossEntropy

Epoch 1/5 | TrainLoss: 1.3071 | ValLoss: 0.9307 | TrainAcc: 62.53% | ValAcc: 71.30%

Epoch 2/5 | TrainLoss: 0.4544 | ValLoss: 0.9741 | TrainAcc: 85.30% | ValAcc: 71.38%

Epoch 3/5 | TrainLoss: 0.2546 | ValLoss: 0.9752 | TrainAcc: 91.36% | ValAcc: 72.01%

Epoch 4/5 | TrainLoss: 0.2061 | ValLoss: 1.0333 | TrainAcc: 92.83% | ValAcc: 71.49%

Epoch 5/5 | TrainLoss: 0.1780 | ValLoss: 0.9909 | TrainAcc: 94.18% | ValAcc: 72.80%
```

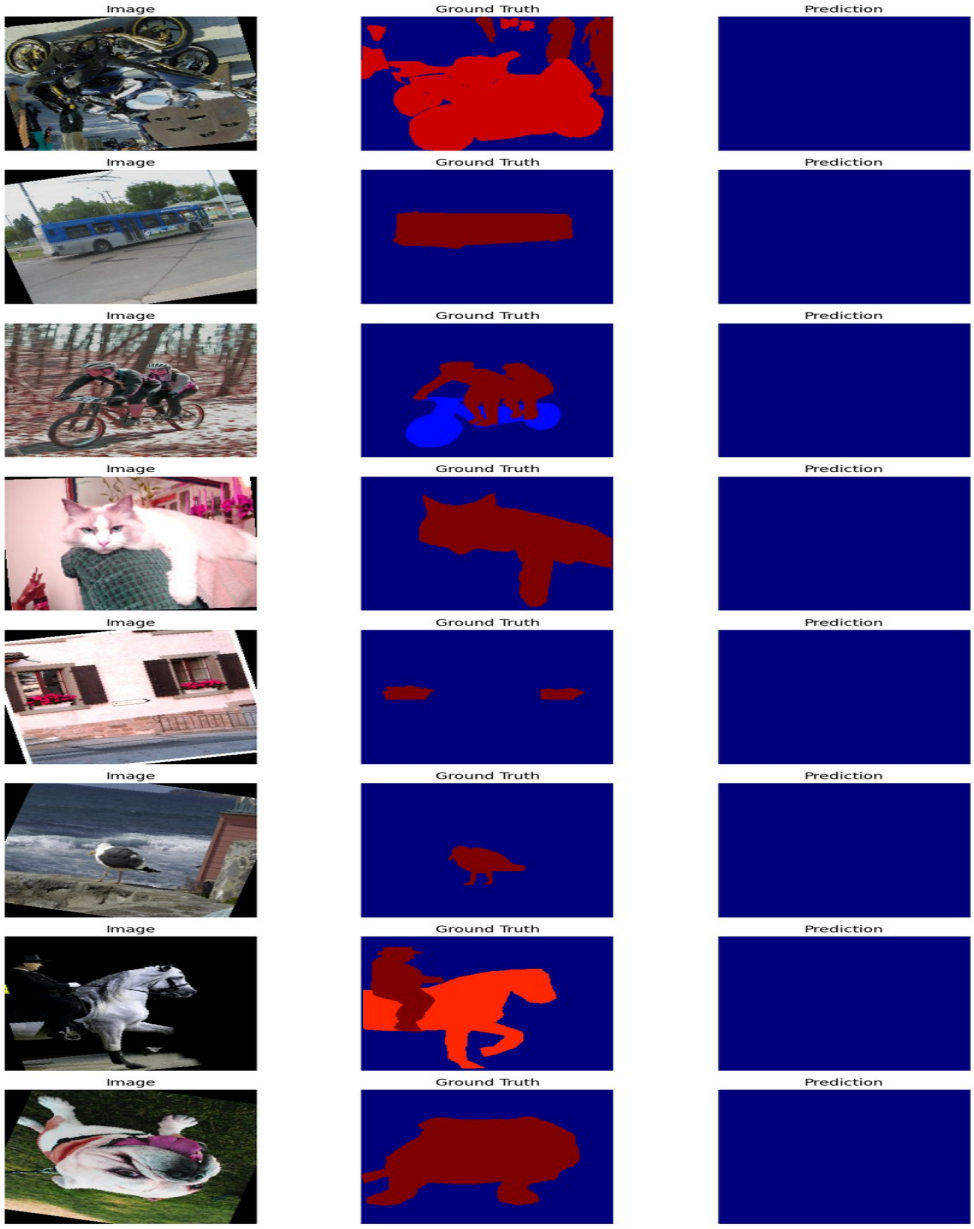


Σχόλιο: Παρατηρώ ότι η συνάρτηση CrossEntropy με label_smoothing=0.1 οδηγεί σε σημαντικά υψηλότερες τιμές της συνάρτησης κόστους, ωστόσο αυτή ενώ ναι μεν μείωσε την αυτοπεποίθηση του μοντέλου, η γενίκευση το οδήγησε και σε χειρότερες προβλέψεις στο validation.



Άσκηση #3

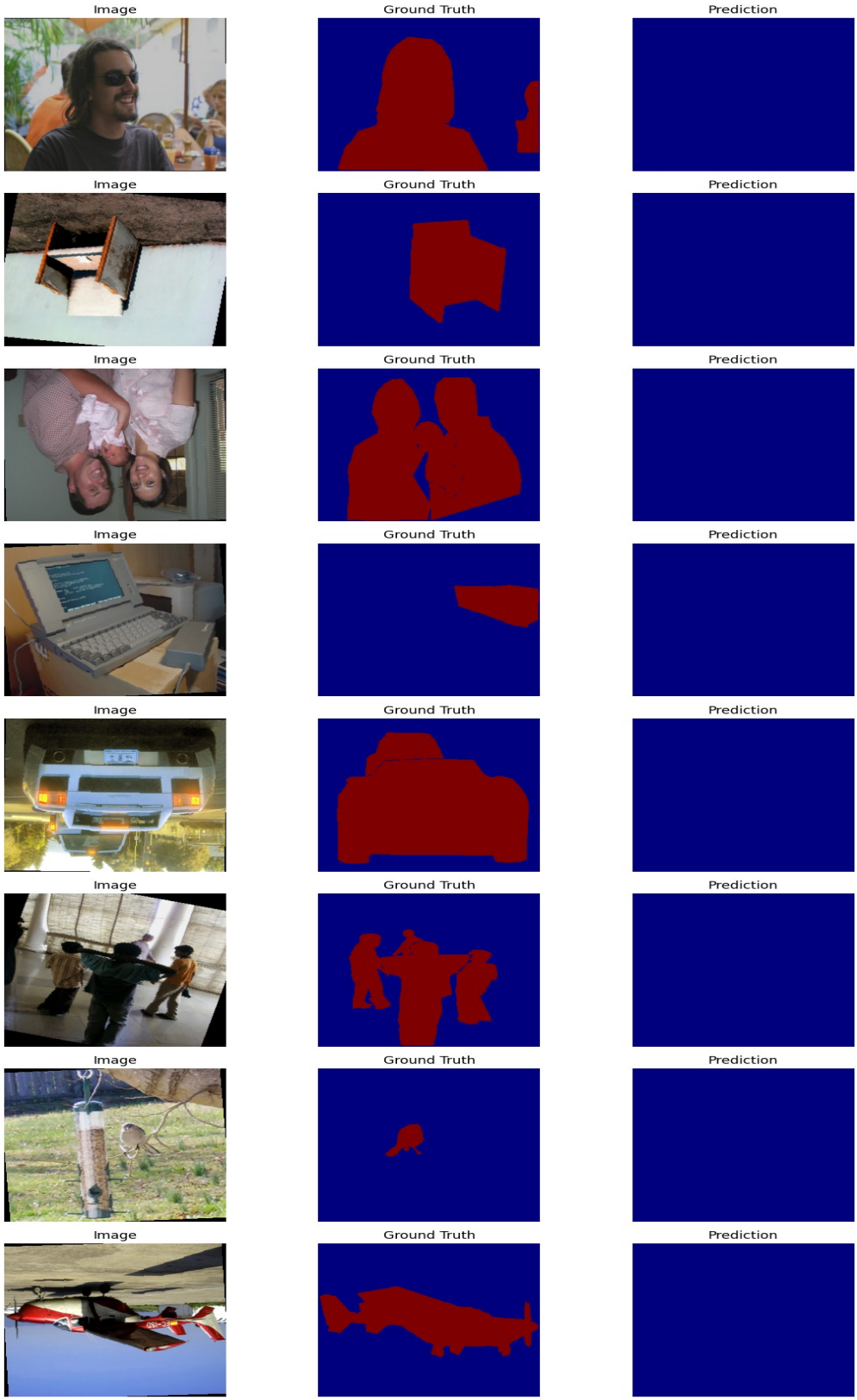
Σε αυτήν την άσκηση για υπολογιστικούς λόγους χρησιμοποίησα subset 4000 εικόνων. Στις δοκιμές που προσπάθησα με τα διαφορετικά epochs και τα διαφορετικά learning rate στα περισσότερα από αυτά έβλεπα να έχουν acurracy γύρω στο 70.47% και νόμιζα ότι το μοντέλο δυσκολεύεται να μάθει περισσότερα και έχει ταβάνι κοντά σε αυτόν τον αριθμό. Ωστόσο το Val IoU ήταν πάντα μικρό και τελικά αυτό που αποκαλύπτεται είναι ότι όταν έτρεχα να δείξει τα prediction σε κάθε εικόνα απλά μάντευε ότι όλα τα pixel είναι background και επειδή το background κατέχει το μεγαλύτερο κομμάτι των pixel έπαιρνε τέτοιο accuracy και δεν μπορούσε να το βελτιώσει περαιτέρω. Για αυτό έκανα διάφορες δοκιμές βάζοντας και βάρη στο background ώστε να μην παίρνει την safe οδό να μαντεύει background. Παρόλο αυτά ότι και να δοκίμασα κατέληγε να έχει πάλι χαμηλό IoU. Αυτό καταλήγω ότι συμβαίνει επειδή το foreground είναι η μειοψηφία των pixel και οι κλάσεις του είναι ανισοκατανεμημένες.
Παράδειγμα 70% accuracy:



```
Training with epochs=3

Epoch 1/3 | Train Loss: 2.3411, Val Loss: 2.3459 | Train Acc: 68.65, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 2/3 | Train Loss: 2.2106, Val Loss: 2.3063 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 3/3 | Train Loss: 2.2017, Val Loss: 2.3044 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842
Training with epochs=6

Epoch 1/6 | Train Loss: 2.3499, Val Loss: 2.3264 | Train Acc: 67.61, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 2/6 | Train Loss: 2.2068, Val Loss: 2.2879 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 3/6 | Train Loss: 2.1969, Val Loss: 2.2909 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 4/6 | Train Loss: 2.1928, Val Loss: 2.2973 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 5/6 | Train Loss: 2.1919, Val Loss: 2.2885 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 6/6 | Train Loss: 2.1901, Val Loss: 2.2774 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842
Training with epochs=9

Epoch 1/9 | Train Loss: 2.3293, Val Loss: 2.3244 | Train Acc: 69.75, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 2/9 | Train Loss: 2.2081, Val Loss: 2.2910 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 3/9 | Train Loss: 2.1996, Val Loss: 2.2843 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 4/9 | Train Loss: 2.1942, Val Loss: 2.2754 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 5/9 | Train Loss: 2.1922, Val Loss: 2.3010 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 6/9 | Train Loss: 2.1905, Val Loss: 2.2724 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 7/9 | Train Loss: 2.1865, Val Loss: 2.2692 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 8/9 | Train Loss: 2.1844, Val Loss: 2.2674 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 9/9 | Train Loss: 2.1831, Val Loss: 2.2743 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842
```

Το ίδιο παρατηρείται και για διαφορετικά learning rate:

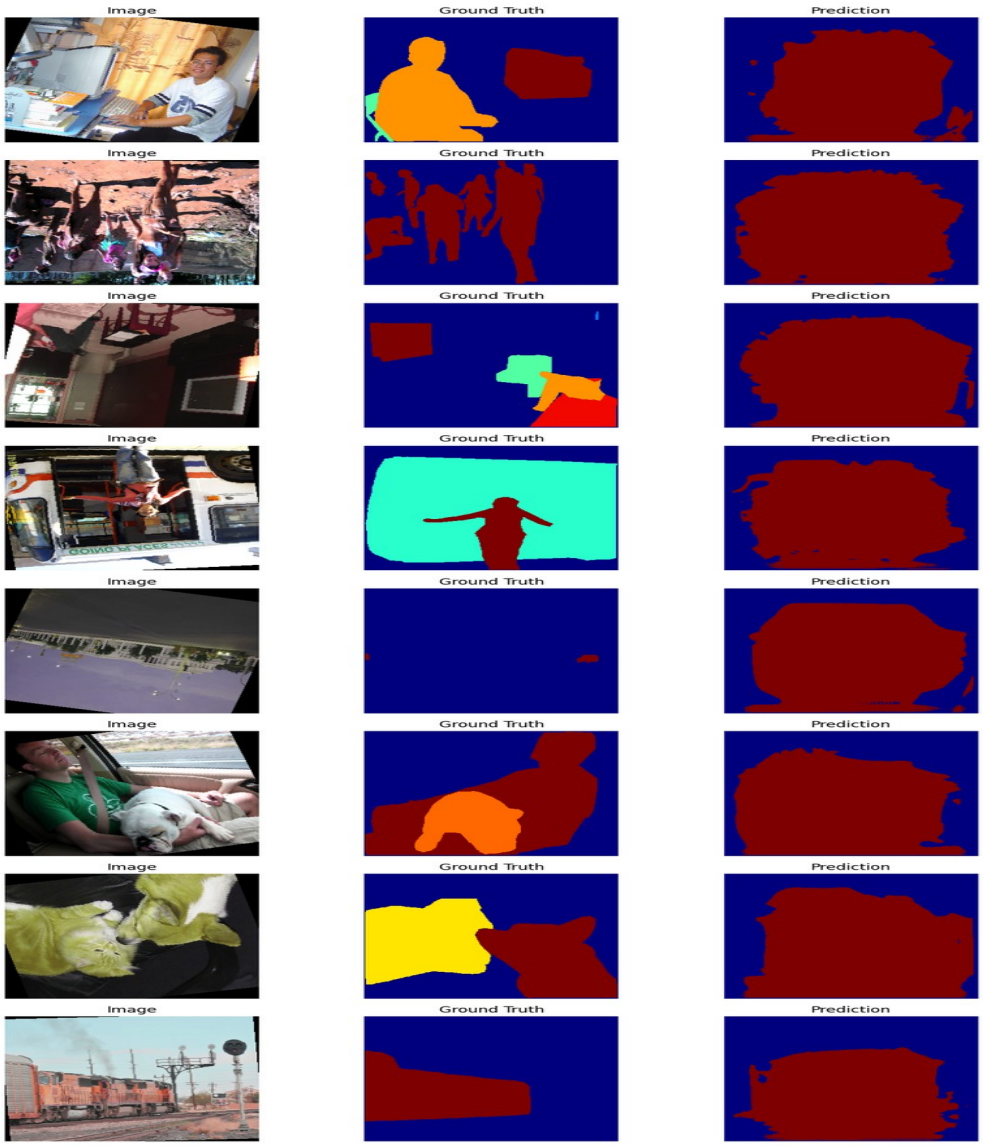
```
Training with lr=0.01

Epoch 1/5 | Train Loss: 11673866985.7589, Val Loss: 2.4085 | Train Acc: 65.95, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 2/5 | Train Loss: 2.2579, Val Loss: 2.3710 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 3/5 | Train Loss: 2.2502, Val Loss: 2.3665 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 4/5 | Train Loss: 2.2455, Val Loss: 2.3552 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 5/5 | Train Loss: 2.2401, Val Loss: 2.3485 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842
Training with lr=0.001

Epoch 1/5 | Train Loss: 2.3351, Val Loss: 2.3689 | Train Acc: 69.63, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 2/5 | Train Loss: 2.2514, Val Loss: 2.3402 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 3/5 | Train Loss: 2.2393, Val Loss: 2.3534 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 4/5 | Train Loss: 2.2322, Val Loss: 2.3270 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 5/5 | Train Loss: 2.2283, Val Loss: 2.3396 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842
Training with lr=0.0001

Epoch 1/5 | Train Loss: 2.3431, Val Loss: 2.3282 | Train Acc: 67.39, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 2/5 | Train Loss: 2.2038, Val Loss: 2.3044 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 3/5 | Train Loss: 2.1976, Val Loss: 2.2835 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 4/5 | Train Loss: 2.1924, Val Loss: 2.2803 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842

Epoch 5/5 | Train Loss: 2.1914, Val Loss: 2.2754 | Train Acc: 70.47, Val Acc: 70.33% | Val IoU: 0.0842
```



Σχόλιο:
Και εδώ φαίνεται ότι δεν είναι θέμα learning rate καθώς το μοντέλο συνεχίζει και απλά μαντεύει ότι όλη φωτογραφία είναι ένα background

Αποτελέσματα με βάρος στο background:

```
Epoch 1/9 | Train Loss: 3.1191, Val Loss: 3.5573 | Train Acc: 16.75, Val Acc: 10.24% | Val IoU: 0.0130
Epoch 2/9 | Train Loss: 3.0627, Val Loss: 3.5260 | Train Acc: 17.79, Val Acc: 10.09% | Val IoU: 0.0128
Epoch 3/9 | Train Loss: 3.0560, Val Loss: 3.4896 | Train Acc: 21.27, Val Acc: 33.17% | Val IoU: 0.0545
Epoch 4/9 | Train Loss: 3.0228, Val Loss: 3.4606 | Train Acc: 47.96, Val Acc: 44.96% | Val IoU: 0.0746
Epoch 5/9 | Train Loss: 3.0084, Val Loss: 3.4847 | Train Acc: 48.58, Val Acc: 26.00% | Val IoU: 0.0417
Epoch 6/9 | Train Loss: 2.9982, Val Loss: 3.4339 | Train Acc: 48.07, Val Acc: 49.01% | Val IoU: 0.0811
Epoch 7/9 | Train Loss: 2.9895, Val Loss: 3.4290 | Train Acc: 47.90, Val Acc: 52.14% | Val IoU: 0.0860
Epoch 8/9 | Train Loss: 2.9918, Val Loss: 3.4301 | Train Acc: 47.89, Val Acc: 48.34% | Val IoU: 0.0803
Epoch 9/9 | Train Loss: 2.9862, Val Loss: 3.4232 | Train Acc: 48.26, Val Acc: 44.03% | Val IoU: 0.0733
```

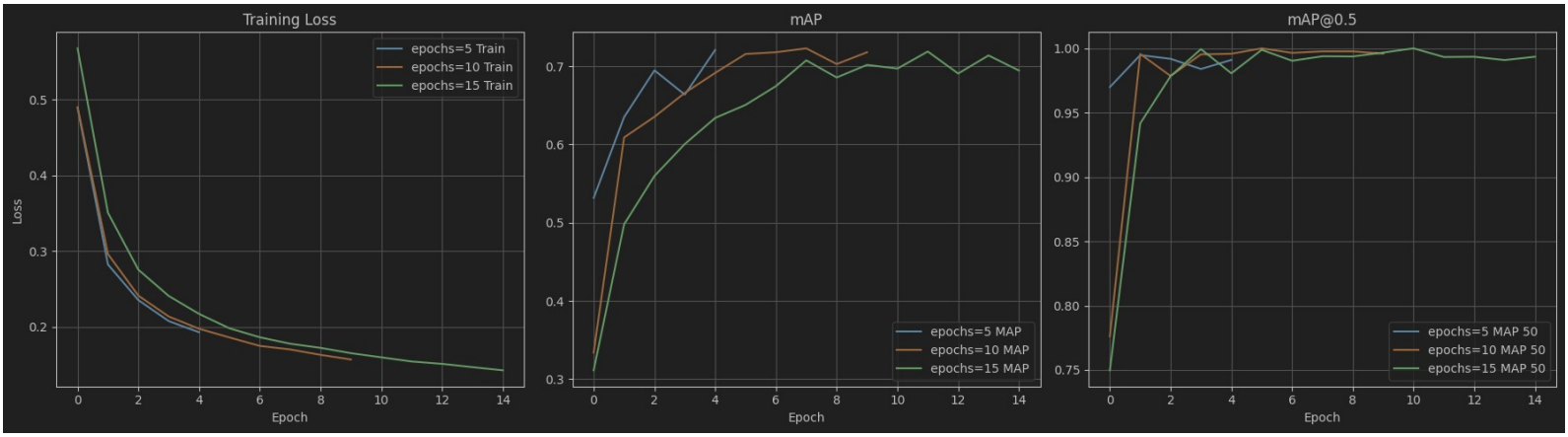


Σχόλιο:
Εδώ με τα βάρη στο background το μοντέλο παίρνει παραπάνω ρίσκο στο να βρίσκει αντικείμενα και αυτό φαίνεται, ωστόσο για τα προβλήματα που ανέλυσα πριν δεν μαθαίνει σωστά.

Άσκηση #4

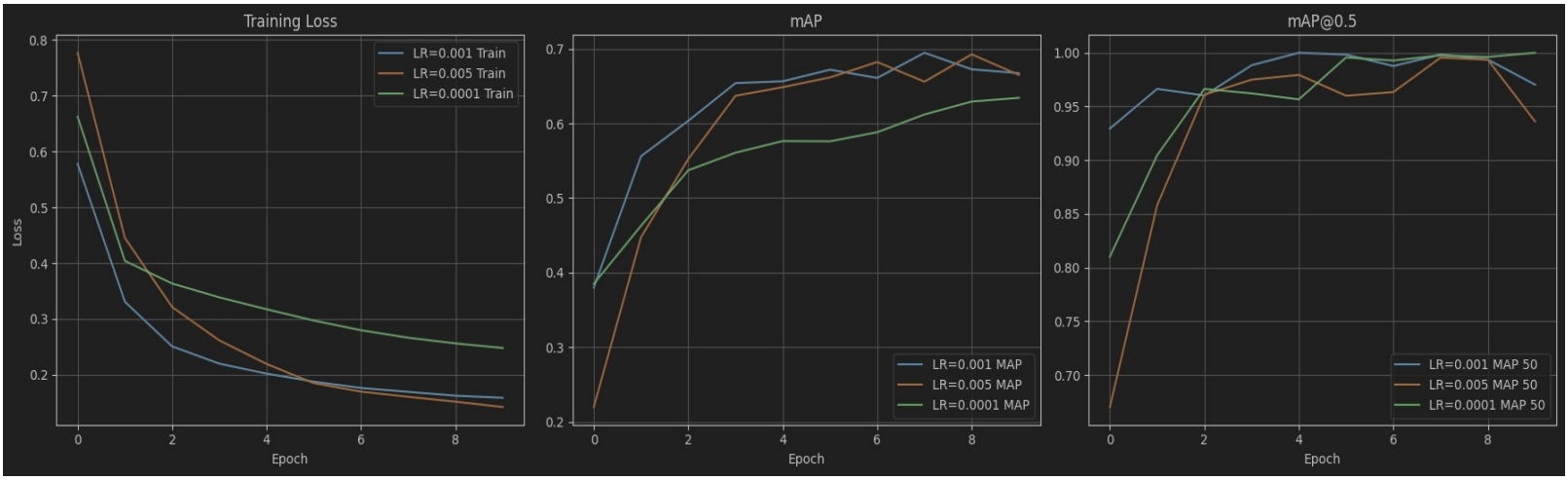
Σε αυτή την άσκηση χρησιμοποίησα subset(300) για training και subset(50) validation λόγω του πολύ χρόνου που έπαιρναν τα πειράματα για να τρέξουν.

Για διαφορετικά **epochs**:



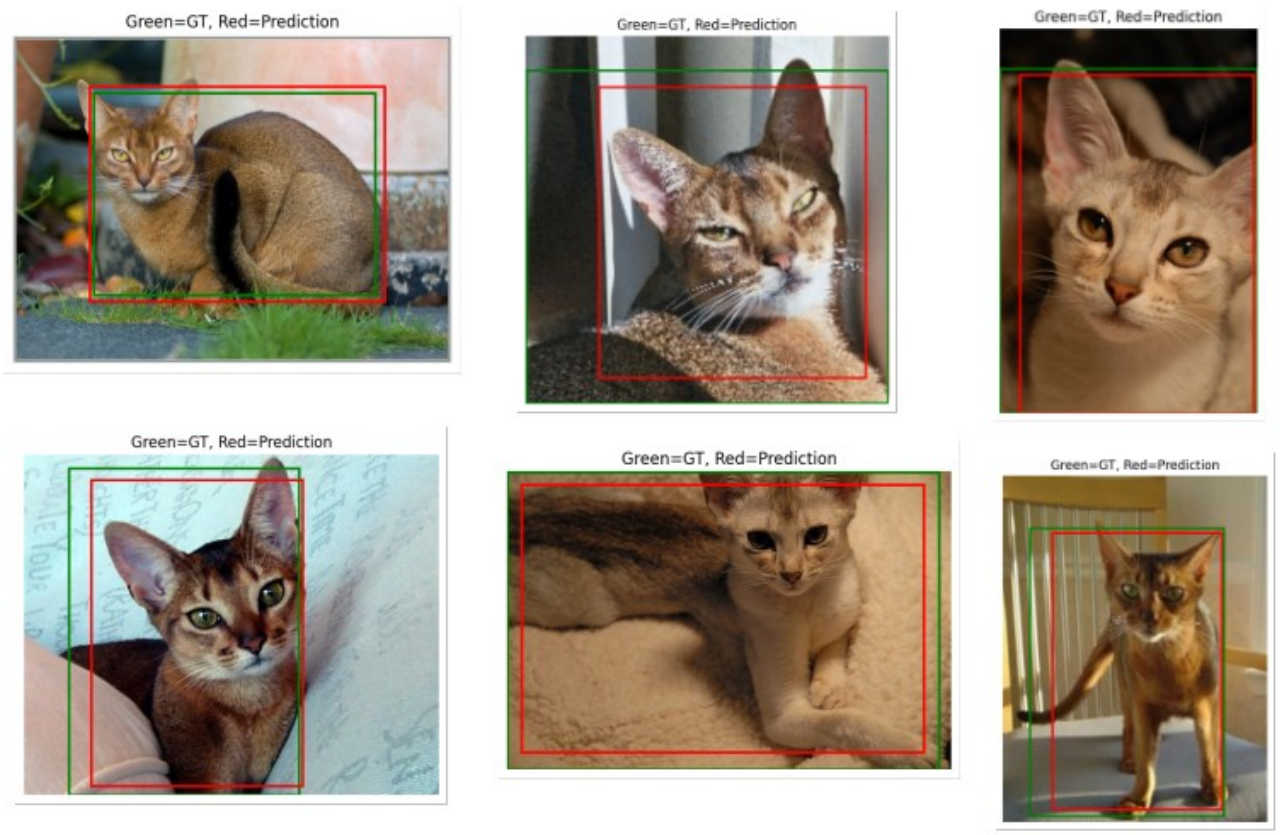
Σχόλιο:
Για τα διαφορετικά epochs πέρα από το **training_loss**, που έχει μία μικρή διαφορά, στα υπόλοιπα δεν υπάρχει κάποια βελτίωση. Αυτό σημαίνει ότι το μοντέλο μαθαίνει καλά το subset από την πρώτο epoch και τα επιπλέον epoch δεν μπορούν να βελτιώσουν πολύ περισσότερο το μοντέλο.

Για διαφορετικά **learning rate**:

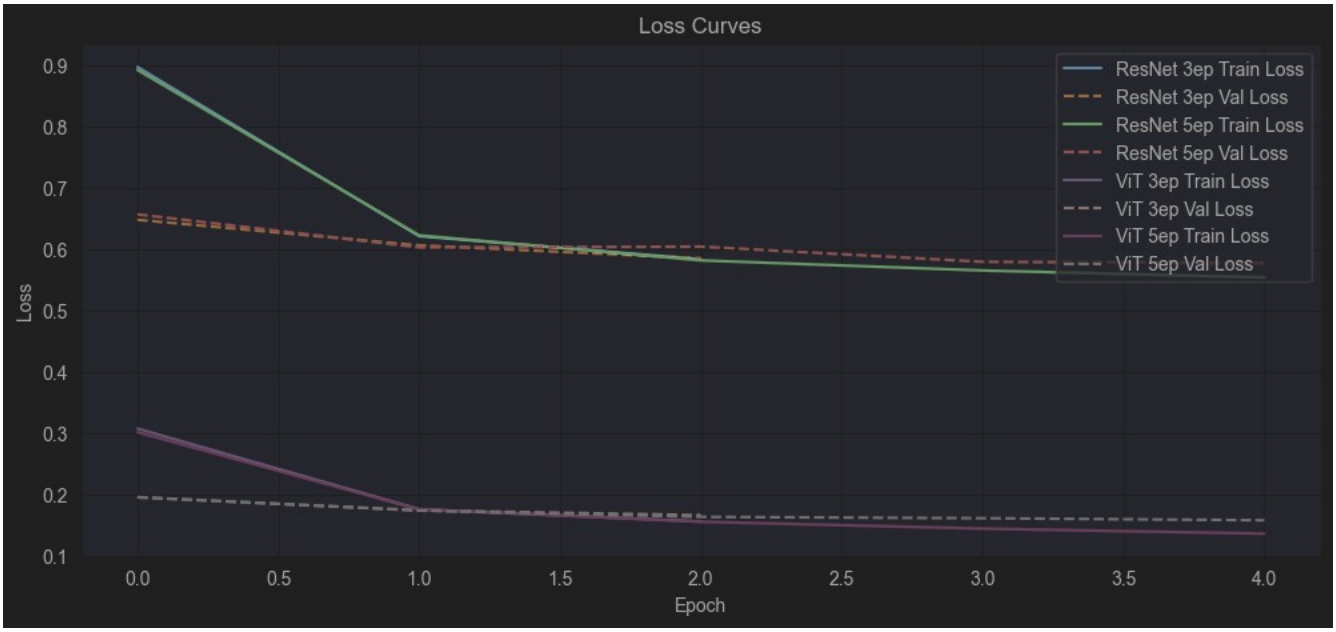


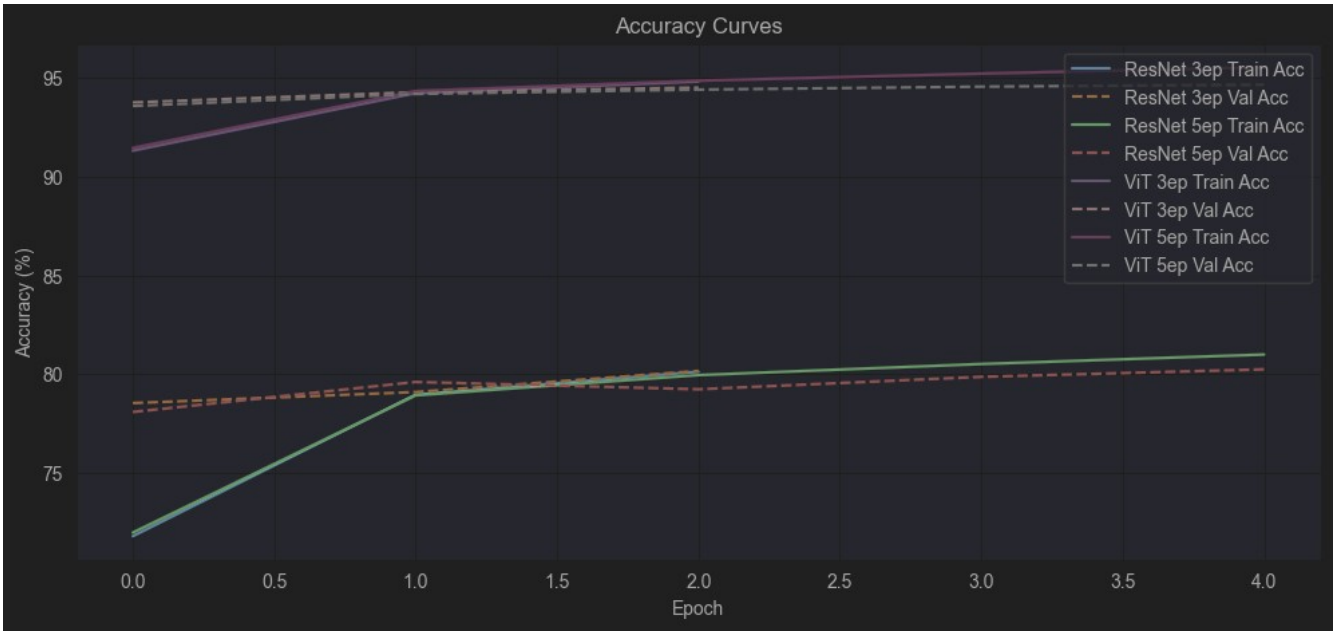
Σχόλιο:
Για learning_rate=0.001 και learning_rate=0.005 τα αποτελέσματα έχουν σχεδόν το ίδια αποτελέσματα με την διαφορά να την κάνει learning_rate=0.0001 όπου έχει σαφώς χαμηλότερα αποτελέσματα. Αυτό πιθανότατα γίνεται επειδή το **MASK R-CNN** που χρησιμοποιώ χρησιμοποιεί pretrained βάρη και λόγω του μικρού learning rate επιβραδύνεται η προσαρμογή των clarification & masks heads. Για αυτό σε περιορισμένο αριθμό από epochs είναι κατώτερη επιλογή.

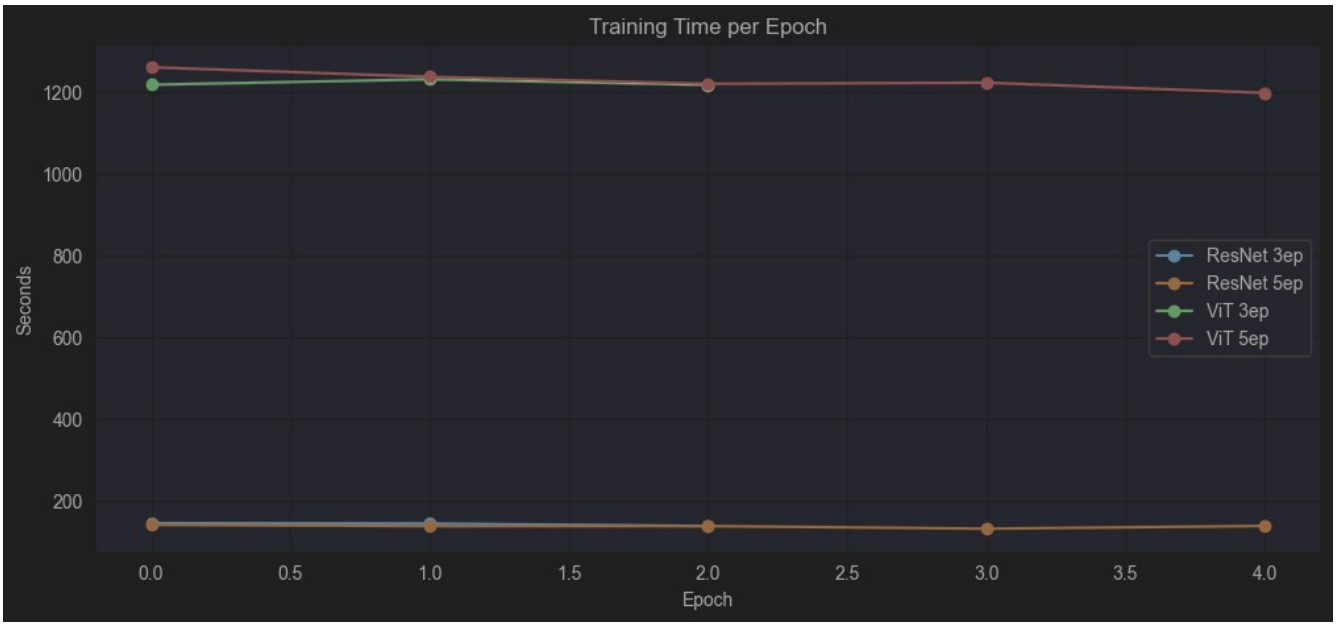
Αποτελέσματα για lr=1e-3 epochs=15:



Άσκηση #5
Για διαφορετικά epochs:


Loss Curves
ResNet 3ep Train Loss
ResNet 3ep Val Loss
ResNet 5ep Train Loss
ResNet 5ep Val Loss
ViT 3ep Train Loss
ViT 3ep Val Loss
ViT 5ep Train Loss
ViT 5ep Val Loss
Loss
Epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

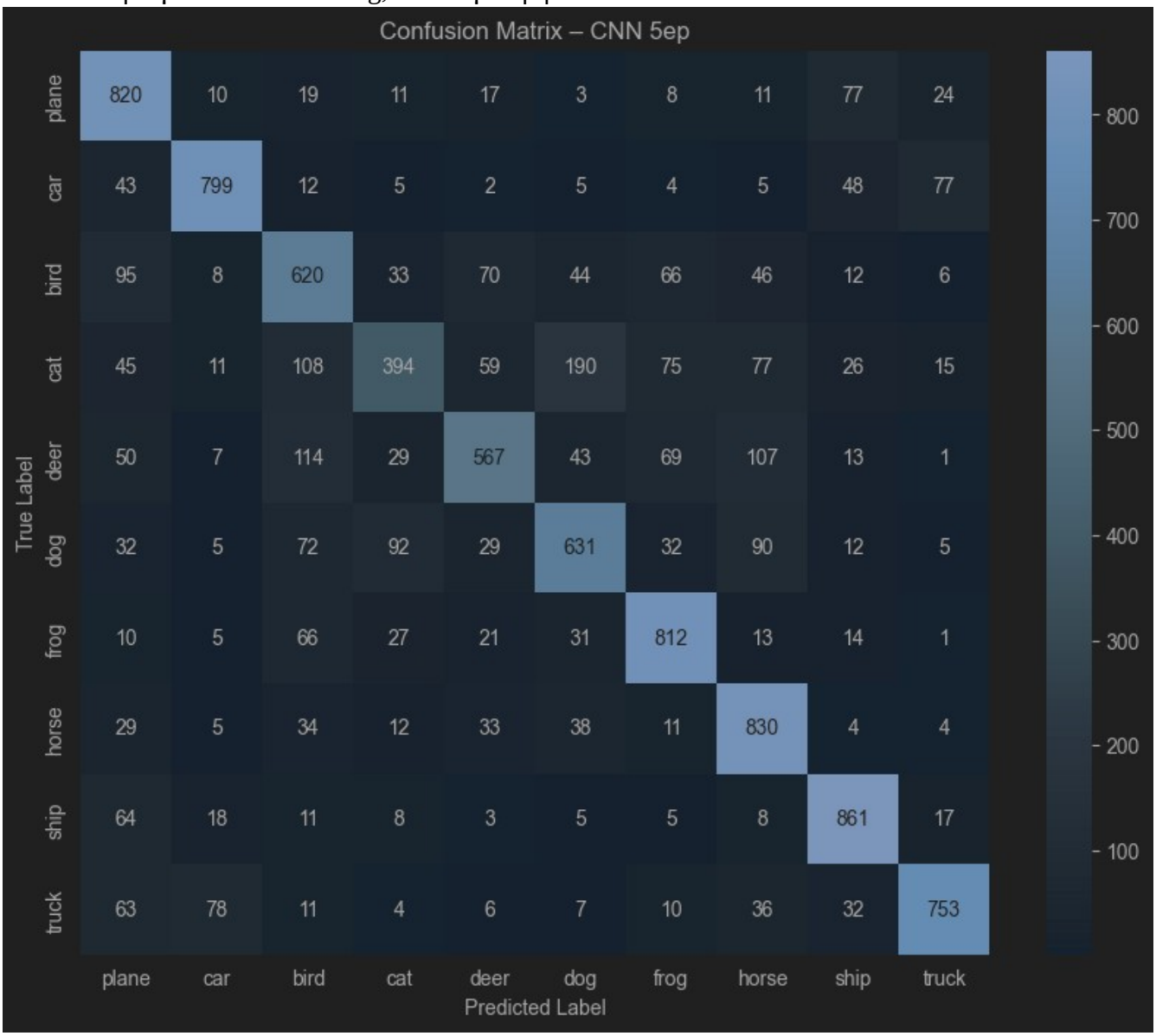

Accuracy Curves
ResNet 3ep Train Acc
ResNet 3ep Val Acc
ResNet 5ep Train Acc
ResNet 5ep Val Acc
ViT 3ep Train Acc
ViT 3ep Val Acc
ViT 5ep Train Acc
ViT 5ep Val Acc
Accuracy (%)
95
90
85
80
75
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

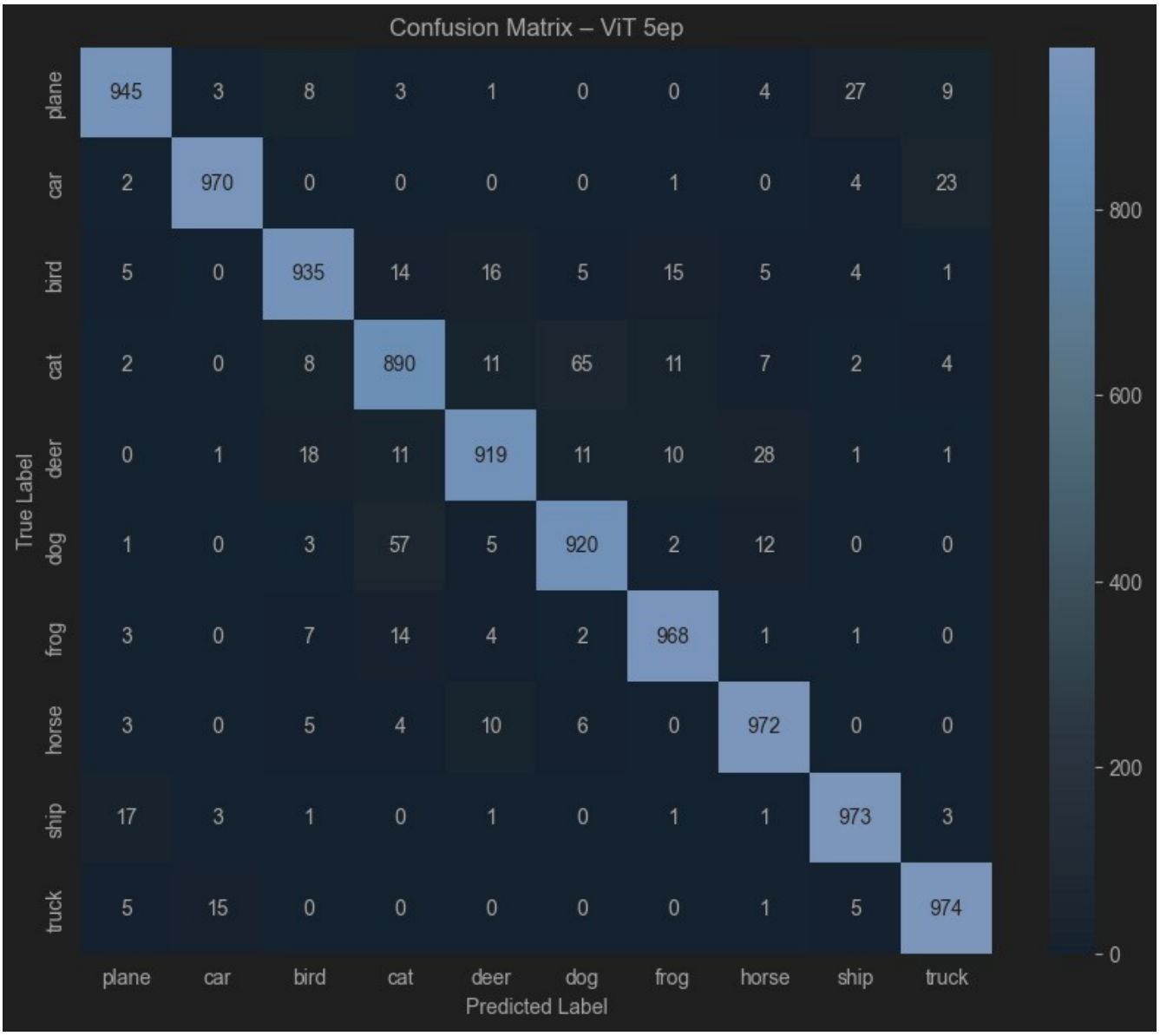

Training Time per Epoch
ResNet 3ep
ResNet 5ep
ViT 3ep
ViT 5ep
Seconds
1200
1000
800
600
400
200
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch


Efficiency: Time Cost per Accuracy Unit
Seconds per 1% Accuracy
60
50
40
30
20
10
0
ResNet 3ep
ResNet 5ep
ViT 3ep
ViT 5ep

```
⏱ Efficiency Table:
ResNet 3ep      : 5.35 seconds per 1% accuracy
ResNet 5ep      : 8.60 seconds per 1% accuracy
ViT 3ep         : 38.77 seconds per 1% accuracy
ViT 5ep         : 64.81 seconds per 1% accuracy
```

Σχόλιο:
Στην 5η άσκηση χρησιμοποίησα pretrained μοντέλα σε ImageNet με frozen backbone με το τελευταίο τους layer να είναι το μόνο που μαθαίνει. Το συνελικτικό δίκτυο που χρησιμοποίησα είναι το ResNet και το δίκτυο μετασχηματιστών είναι το ViT. Αυτό που παρατηρείται είναι ότι το ViT έχει σαφώς καλύτερη ακρίβεια ενώ ταυτόχρονα κάτι που παρατηρείται είναι ότι τα **accuracy** και τα **loss** δεν παρουσιάζουν ιδιαίτερη βελτίωση με τα αποτελέσματα μετά απο 3 και 5 epochs να σχεδόν ίδια με το 1ο. Αυτό συμβαίνει λόγω του ότι τα μοντέλα είναι pretrained και έχουν frozen backbone, έτσι ο linear header προσαρμόζεται από το πρώτο epoch σχεδόν εξ ολοκλήρου στα features και μετά δεν μπορεί να βελτιώσει παραπάνω λόγω του frozen backbone. Άρα τα features του ViT είναι πιο διακριτά linearly και καλύτερα για transfer learning, στο συγκεκριμένο dataset.





* Το CNN είναι ResNet, όχι CNN

Σχόλιο:

Αυτό που παρατηρείται στο confusion matrix και στα δύο μοντέλα είναι ότι οι πιο σύνηθες κλάσεις που μπερδεύονται είναι οι:

- σκύλος/γάτα
- ελάφι/άλογο
- αυτοκίνητο/φορτηγό
- αεροπλάνο/πουλί

Κατηγορίες που είναι φυσιολογικό να μπερδεύεται λόγω των ομοιοτήτων σε σχήματα και χαρακτηριστικά που έχουν.

Για διαφορετικά learning rate:

Για learning rate = 1e-3:

```
Epoch 1/3 | Train Loss: 0.8966, Train Acc: 71.79% | Val Loss: 0.6475, Val Acc: 78.53%⏱ Time: 145.6s

Epoch 2/3 | Train Loss: 0.6209, Train Acc: 78.96% | Val Loss: 0.6059, Val Acc: 79.09%⏱ Time: 144.7s

Epoch 3/3 | Train Loss: 0.5833, Train Acc: 80.12% | Val Loss: 0.5848, Val Acc: 80.16%⏱ Time: 138.8s
 Total training time for ResNet 3ep: 7.15 minutes

Epoch 1/3 | Train Loss: 0.3073, Train Acc: 91.31% | Val Loss: 0.1953, Val Acc: 93.76%⏱ Time: 1217.2s

Epoch 2/3 | Train Loss: 0.1760, Train Acc: 94.23% | Val Loss: 0.1741, Val Acc: 94.28%⏱ Time: 1230.3s

Epoch 3/3 | Train Loss: 0.1560, Train Acc: 94.82% | Val Loss: 0.1662, Val Acc: 94.51%⏱ Time: 1216.9s
 Total training time for ViT 3ep: 61.07 minutes
```

Για learning rate = 4e-3:

```
Epoch 1/3 | Train Loss: 1.2407, Train Acc: 63.96% | Val Loss: 0.8471, Val Acc: 75.17%⏱ Time: 136.8s

Epoch 2/3 | Train Loss: 0.7652, Train Acc: 76.29% | Val Loss: 0.7002, Val Acc: 77.55%⏱ Time: 146.7s

Epoch 3/3 | Train Loss: 0.6752, Train Acc: 78.22% | Val Loss: 0.6503, Val Acc: 78.56%⏱ Time: 146.8s
 Total training time for ResNet 5ep (lr=1e-4): 7.17 minutes

Epoch 1/3 | Train Loss: 0.5406, Train Acc: 87.44% | Val Loss: 0.2631, Val Acc: 92.74%⏱ Time: 1259.9s

Epoch 2/3 | Train Loss: 0.2325, Train Acc: 92.93% | Val Loss: 0.2151, Val Acc: 93.38%⏱ Time: 1254.9s

Epoch 3/3 | Train Loss: 0.1995, Train Acc: 93.66% | Val Loss: 0.1957, Val Acc: 93.69%⏱ Time: 1252.8s
 Total training time for ViT 5ep (lr=1e-4): 62.79 minutes
```

Σχόλιο:
Οι διαφορές στο **accuracy** και στο **loss** που παρατηρήθηκαν πριν παρατηρούνται μετά και την αλλαγή του learning rate που εμφανίζει ελαφρώς χειρότερα αποτελέσματα. Αυτό δείχνει ότι όντως τα συμπεράσματα για την ανωτερότητα του ViT που φάνηκαν και νωρίτερα ισχύουν και εδω.

(Επειδή σε αυτήν την άσκηση υπάρχουν πολλά διαγράμματα τα έχω ανεβάσει και στο notebook στο github repo)