Ασκήσεις #1, #2, #3, #4

# Επεξεργασία Εικόνας με Βαθιά Μάθηση

CNN, Transfer Learning, Semantic Segmentation, Object Detection

# Άσκηση #1: Βελτιστοποίηση Αρχιτεκτονικής και Υπερ-παραμέτρων

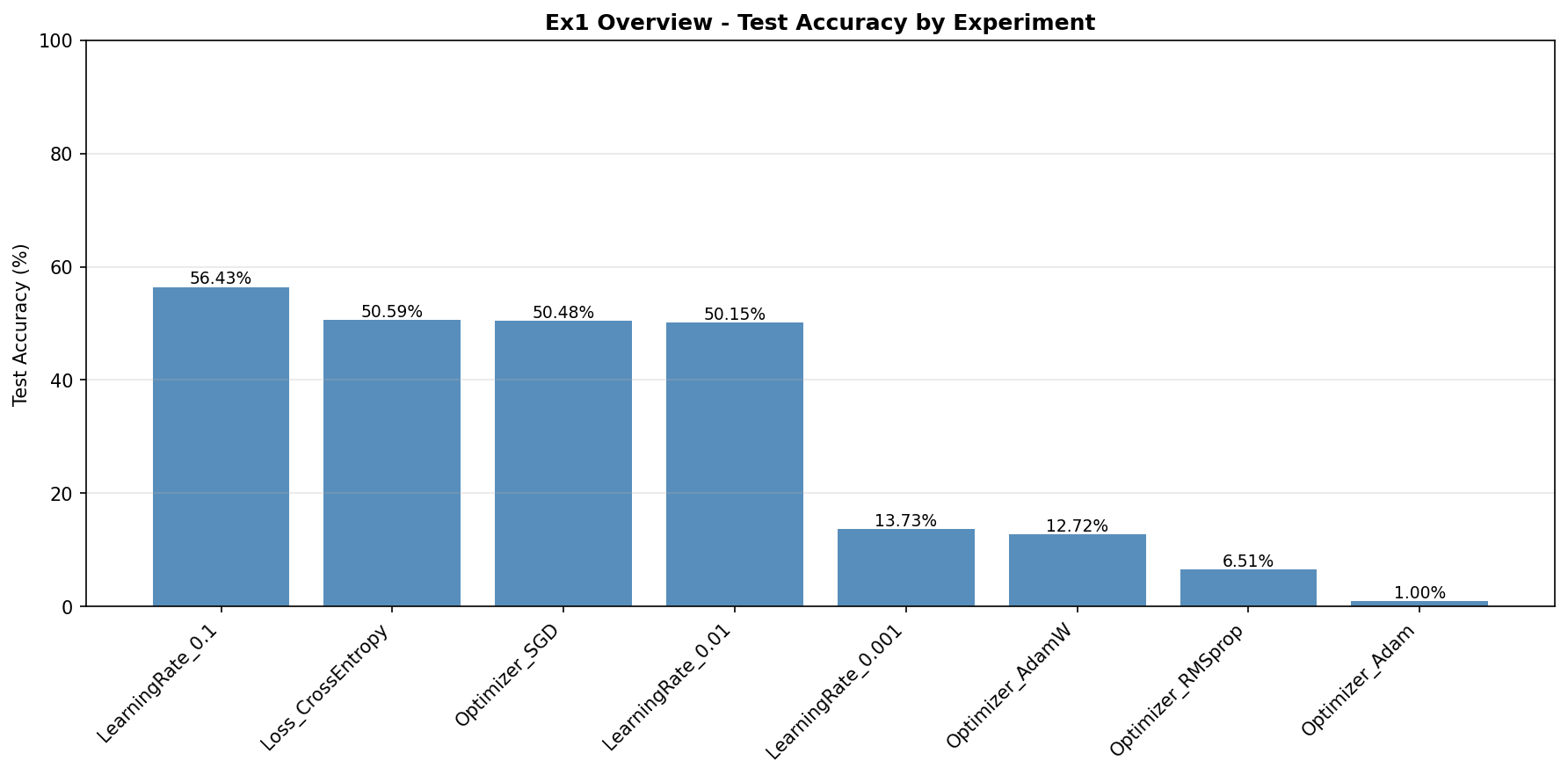
## 1.1 Εισαγωγή

Υλοποιήθηκε ένα Vanilla CNN με 4 συνελικτικά στρώματα (964.516 παράμετροι) για ταξινόμηση στο CIFAR-100 (100 κλάσεις, 32x32 pixels). Το backbone αποτελείται από Conv2d → BatchNorm → ReLU → MaxPool blocks, ακολουθούμενο από fully connected classifier. Εκτελέστηκαν 8 πειράματα σε 3 σειρές (30 epochs, cosine scheduler, batch size 128):

* Σειρά 1 — Συνάρτηση κόστους: CrossEntropy (baseline: SGD, lr=0.01)
* Σειρά 2 — Optimizers: SGD, Adam, AdamW, RMSprop (όλοι με lr=0.01)
* Σειρά 3 — Learning rates: 0.001, 0.01, 0.1 (όλα με SGD)

## 1.2 Συνοπτικός Πίνακας Αποτελεσμάτων

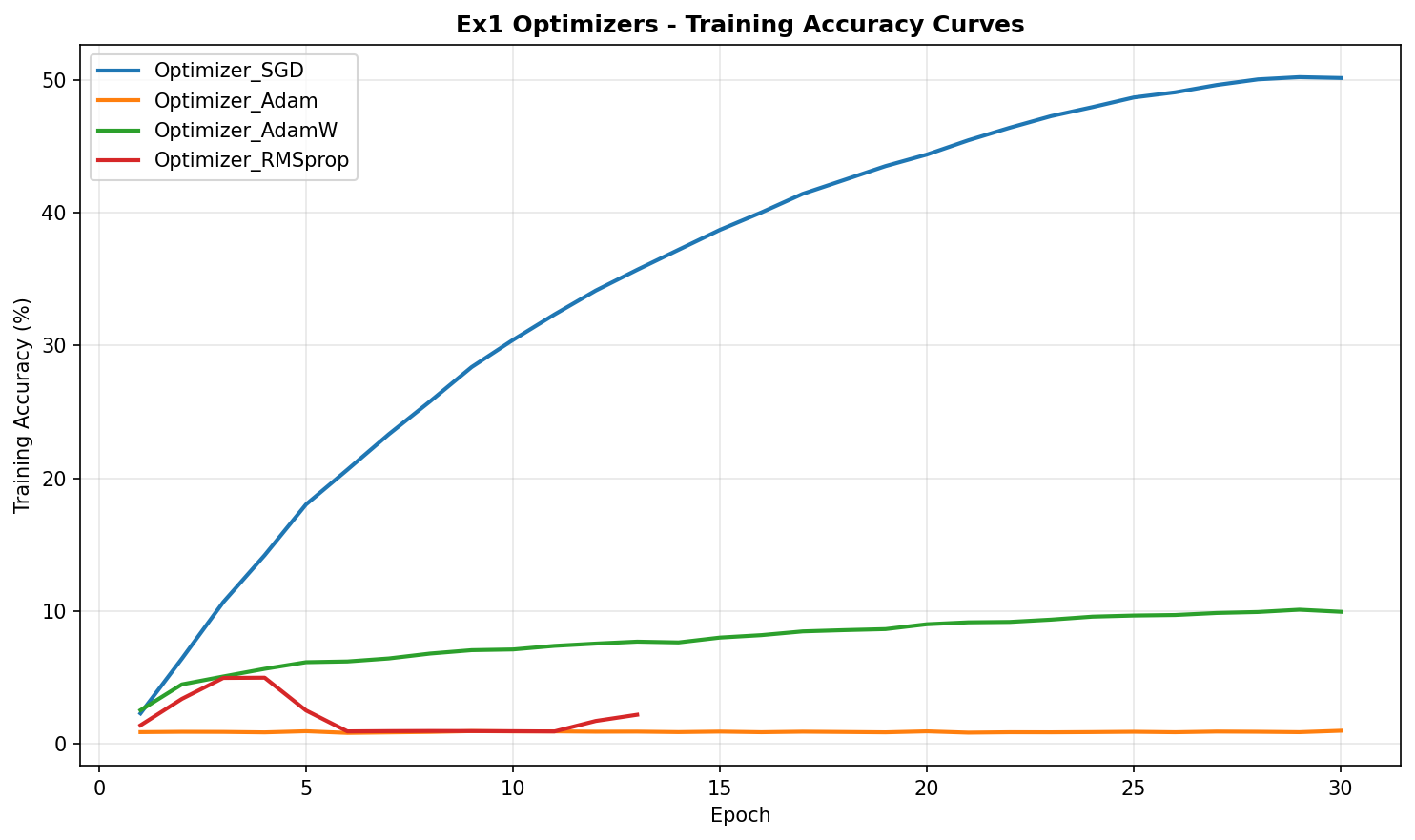
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Πείραμα** | **Optimizer** | **LR** | **Epochs** | **Train(%)** | **Test(%)** | **Χρόνος(min)** |
| LearningRate\_0.1 | SGD | 0.1 | 30 | 59.3 | 56.43 | 6.5 |
| Loss\_CrossEntropy | SGD | 0.01 | 30 | 50.6 | 50.59 | 6.4 |
| Optimizer\_SGD | SGD | 0.01 | 30 | 50.2 | 50.48 | 6.5 |
| LearningRate\_0.01 | SGD | 0.01 | 30 | 50.1 | 50.15 | 6.6 |
| LearningRate\_0.001 | SGD | 0.001 | 30 | 11.7 | 13.73 | 7.2 |
| Optimizer\_AdamW | ADAMW | 0.01 | 30 | 10.1 | 12.72 | 6.4 |
| Optimizer\_RMSprop | RMSPROP | 0.01 | 13 | 5.0 | 6.51 | 2.8 |
| Optimizer\_Adam | ADAM | 0.01 | 30 | 1.0 | 1.00 | 6.6 |



*Εικόνα 1.1: Επισκόπηση ακρίβειας όλων των πειραμάτων*

## 1.3 Σύγκριση Optimizers

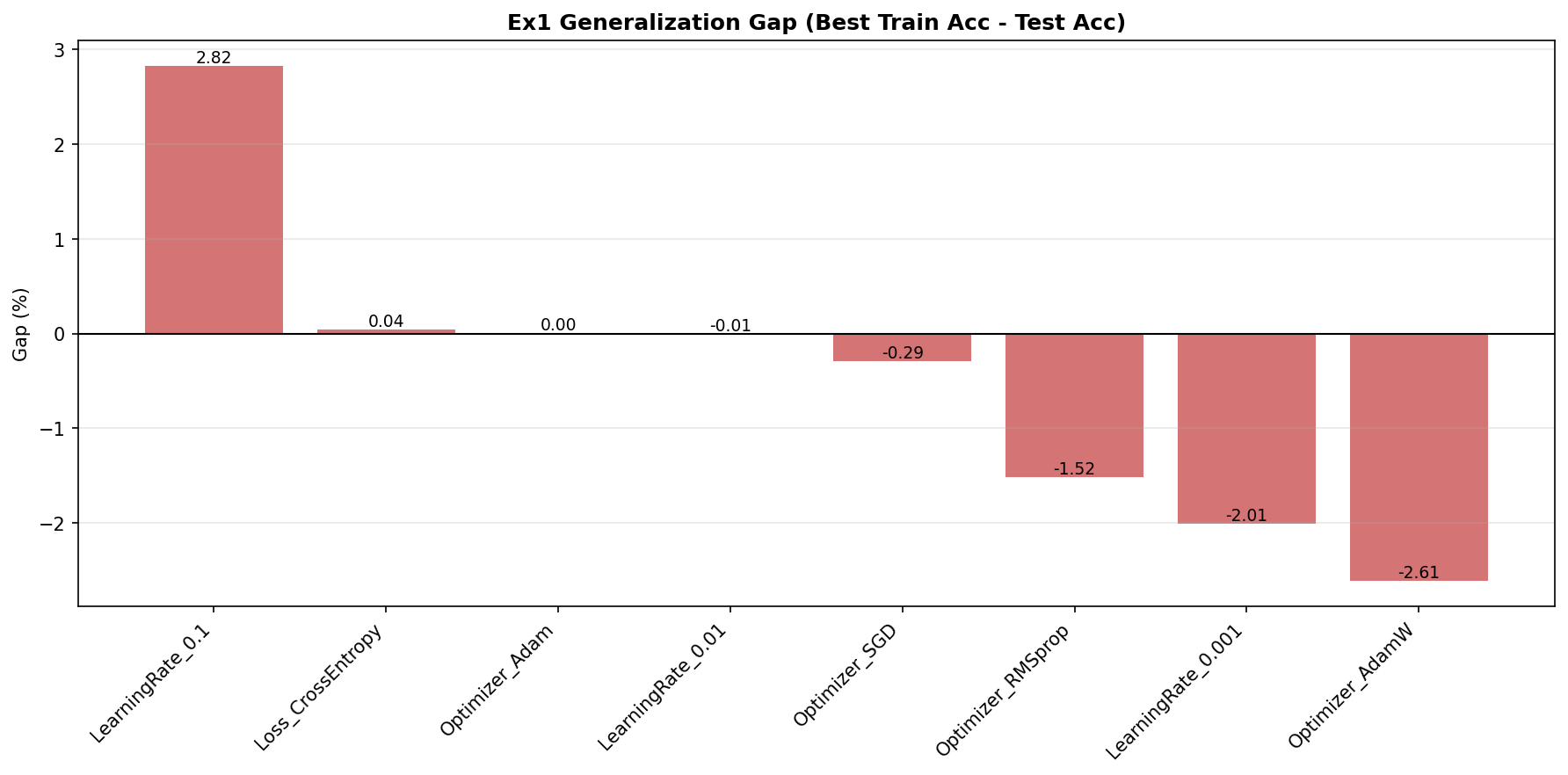
Όλοι οι optimizers δοκιμάστηκαν με lr=0.01. Τα αποτελέσματα δείχνουν ότι η επιλογή optimizer έχει τεράστια επίδραση στην απόδοση:



*Εικόνα 1.2: Καμπύλες εκπαίδευσης ανά optimizer*

Ανάλυση:

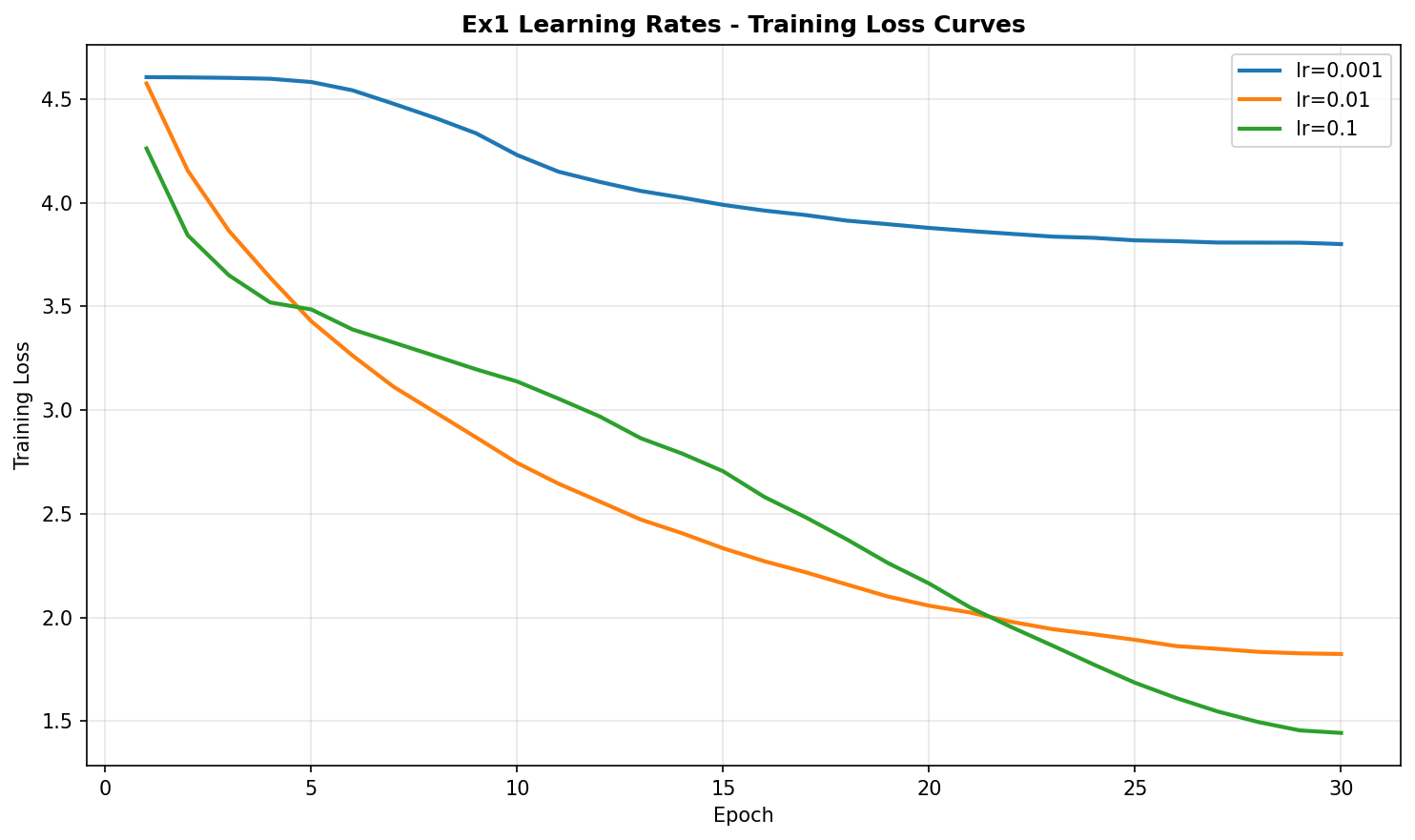
* SGD (50.48%): Καλύτερος optimizer με lr=0.01. Ο SGD χρησιμοποιεί σταθερό gradient scaling, που σε συνδυασμό με το cosine scheduler επιτρέπει ομαλή σύγκλιση. Η καμπύλη ακρίβειας ανεβαίνει σταθερά σε όλα τα 30 epochs.
* AdamW (12.72%): Αποτυχία λόγω υψηλού lr. Οι adaptive optimizers (όπως Adam/AdamW) προσαρμόζουν το LR ανά παράμετρο, άρα με lr=0.01 το πραγματικό LR γίνεται πολύ υψηλό για ένα μικρό CNN, προκαλώντας αστάθεια και αργή σύγκλιση.
* Adam (1.0%): Πλήρης αποτυχία — το loss παρέμεινε σταθερό (~4.605) για 30 epochs. Με lr=0.01 ο Adam υπερπήδησε το ελάχιστο και παγιδεύτηκε σε τυχαία πρόβλεψη (1% = τυχαίο για 100 κλάσεις). Αυτό δείχνει ότι ο Adam απαιτεί lr ≤ 0.001 για μικρά CNNs.
* RMSprop (6.51%): Ασταθής εκπαίδευση με έντονες διακυμάνσεις στο loss (το πρώτο epoch είχε loss=5355!). Έκανε early stop στα 13 epochs. Ο RMSprop δεν χρησιμοποιεί momentum και είναι ευαίσθητος σε υψηλό lr.



*Εικόνα 1.3: Generalization gap (train vs test accuracy)*

## 1.4 Επίδραση Learning Rate

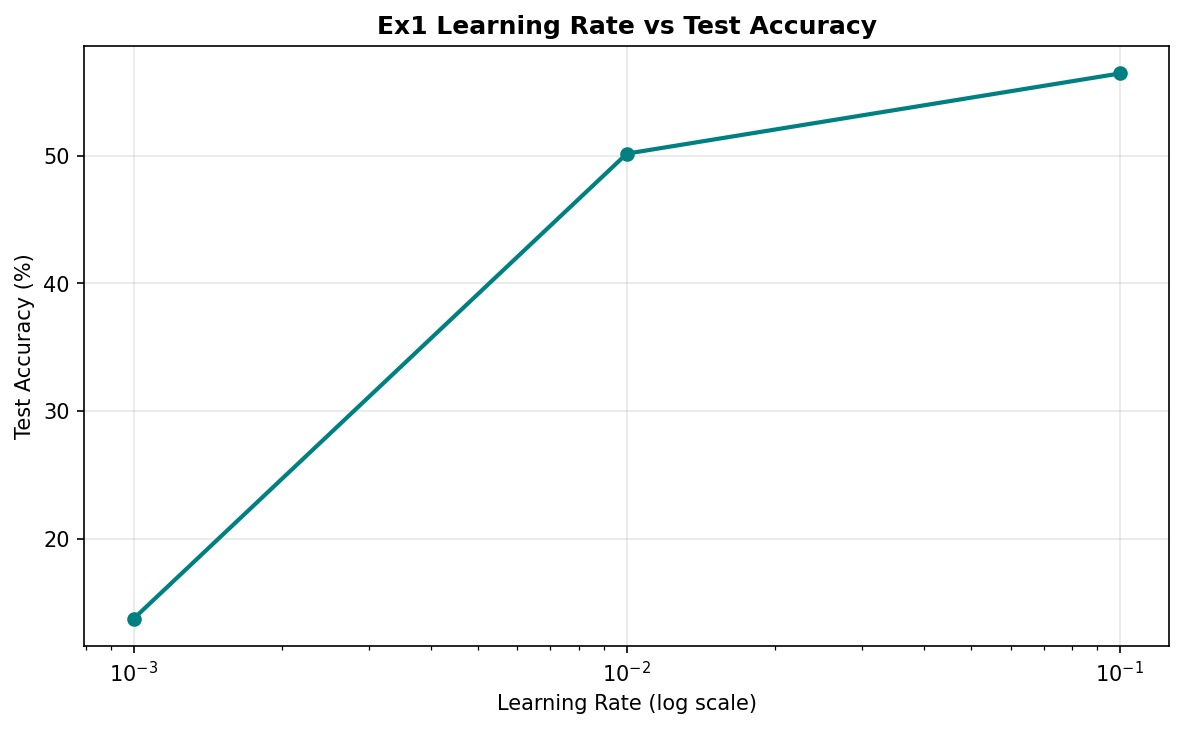
Η σύγκριση τριών learning rates με SGD δείχνει ότι το LR είναι η πιο κρίσιμη υπερ-παράμετρος:



*Εικόνα 1.4: Καμπύλες training loss ανά learning rate*

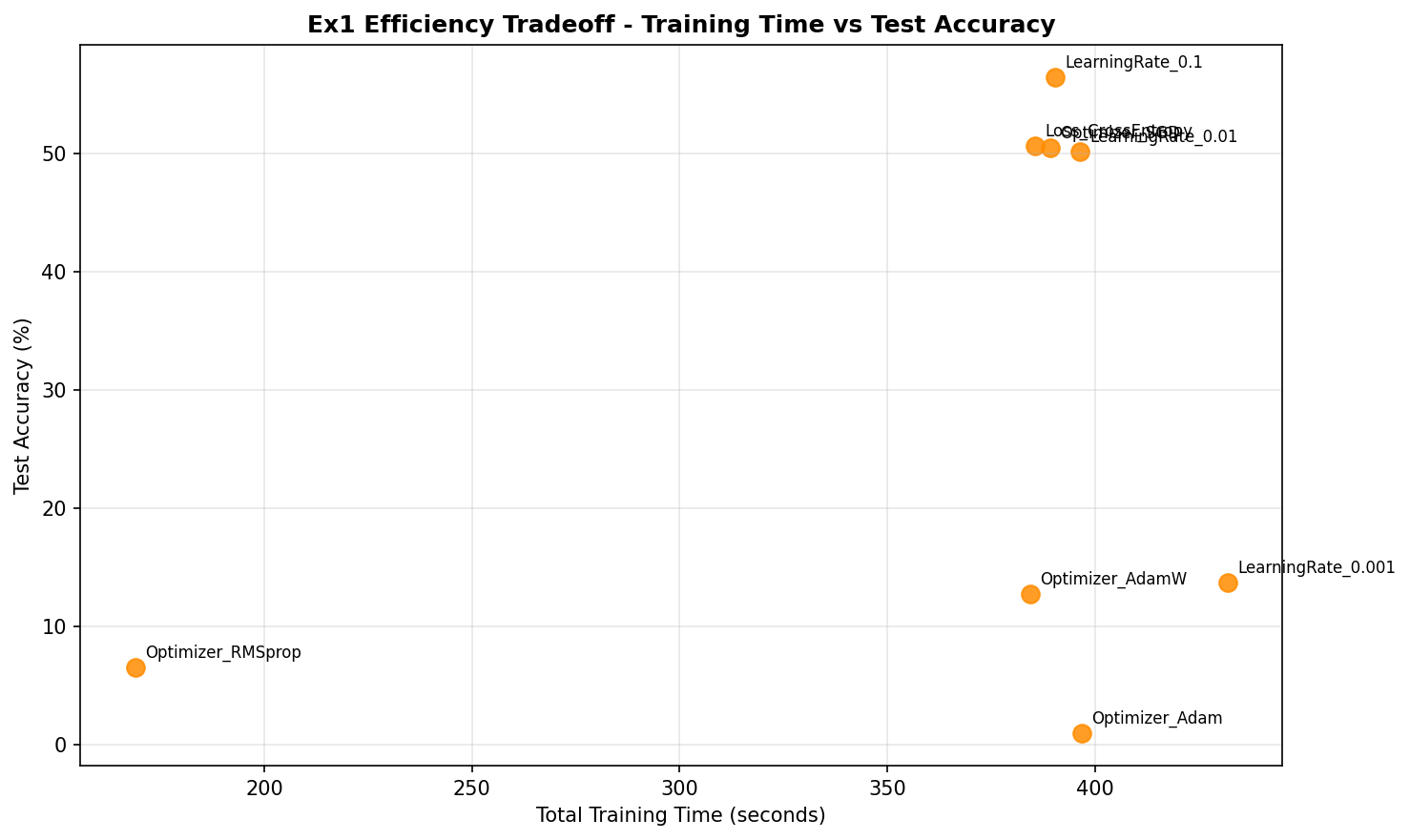
Ανάλυση:

* lr=0.1 (56.43%): Καλύτερο αποτέλεσμα. Το υψηλό lr επιτρέπει γρήγορη εξερεύνηση του loss landscape, και ο cosine scheduler μειώνει σταδιακά το LR για fine-tuning. Στο CIFAR-100 με μικρό CNN, το υψηλό lr βοηθά το μοντέλο να ξεφύγει από τοπικά ελάχιστα.
* lr=0.01 (50.15%): Καλή σύγκλιση αλλά 6% χαμηλότερα. Το μοντέλο συγκλίνει σταθερά αλλά δεν μπορεί να εξερευνήσει αρκετά το loss landscape στα πρώτα epochs.
* lr=0.001 (13.73%): Πολύ αργή σύγκλιση. Σε 30 epochs το loss έπεσε μόλις από 4.60 σε 3.80. Θα χρειαζόταν 100+ epochs για να φτάσει το επίπεδο του lr=0.01.



*Εικόνα 1.5: Learning rate vs test accuracy*

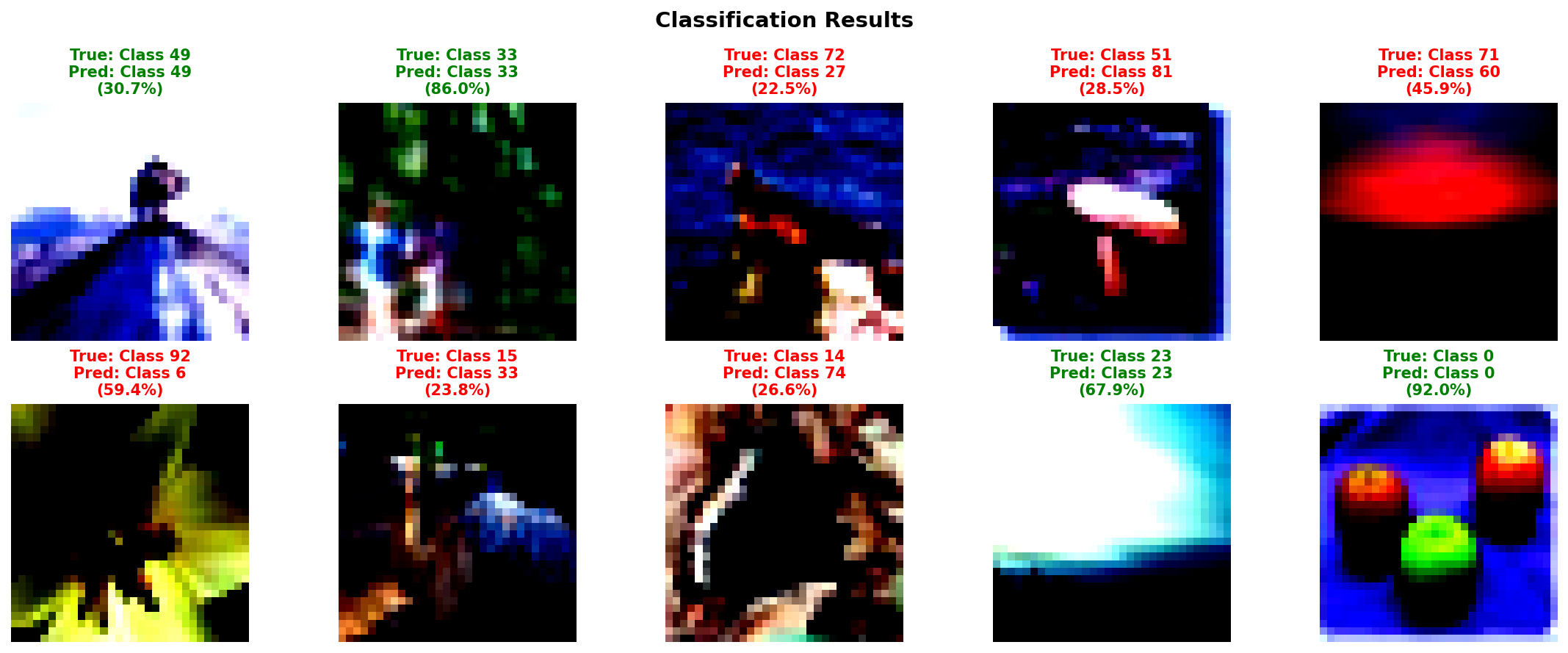
## 1.5 Αποδοτικότητα Χρόνου



*Εικόνα 1.6: Χρόνος εκπαίδευσης vs test accuracy*

Όλα τα πειράματα διήρκεσαν 6-7 λεπτά, εκτός από τον RMSprop (~2.8 min) που έκανε early stop. Ο χρόνος δεν διαφέρει σημαντικά μεταξύ optimizers, άρα η επιλογή βασίζεται καθαρά στην ακρίβεια.

## 1.6 Ενδεικτικά Αποτελέσματα Ταξινόμησης



*Εικόνα 1.7: Δείγματα προβλέψεων — Καλύτερο μοντέλο (SGD, lr=0.1)*

## 1.7 Συμπεράσματα Άσκησης 1

* Ο SGD είναι ο καταλληλότερος optimizer για μικρά CNNs, διότι δεν προσαρμόζει το LR αυτόματα και επωφελείται από τον cosine scheduler.
* Οι adaptive optimizers (Adam, AdamW, RMSprop) απαιτούν πολύ χαμηλότερο lr (≤ 0.001) για να μην αποσταθεροποιήσουν την εκπαίδευση.
* Το learning rate είναι η πιο κρίσιμη υπερ-παράμετρος: από 1% (lr=0.01+Adam) έως 56.43% (lr=0.1+SGD), δηλαδή διαφορά 55 μονάδων.
* Το 56.43% είναι αξιοπρεπές για ένα vanilla CNN χωρίς data augmentation, residual connections ή pretrained weights σε 100 κλάσεις.

# Άσκηση #2: Μεταφορά Μάθησης (Transfer Learning)

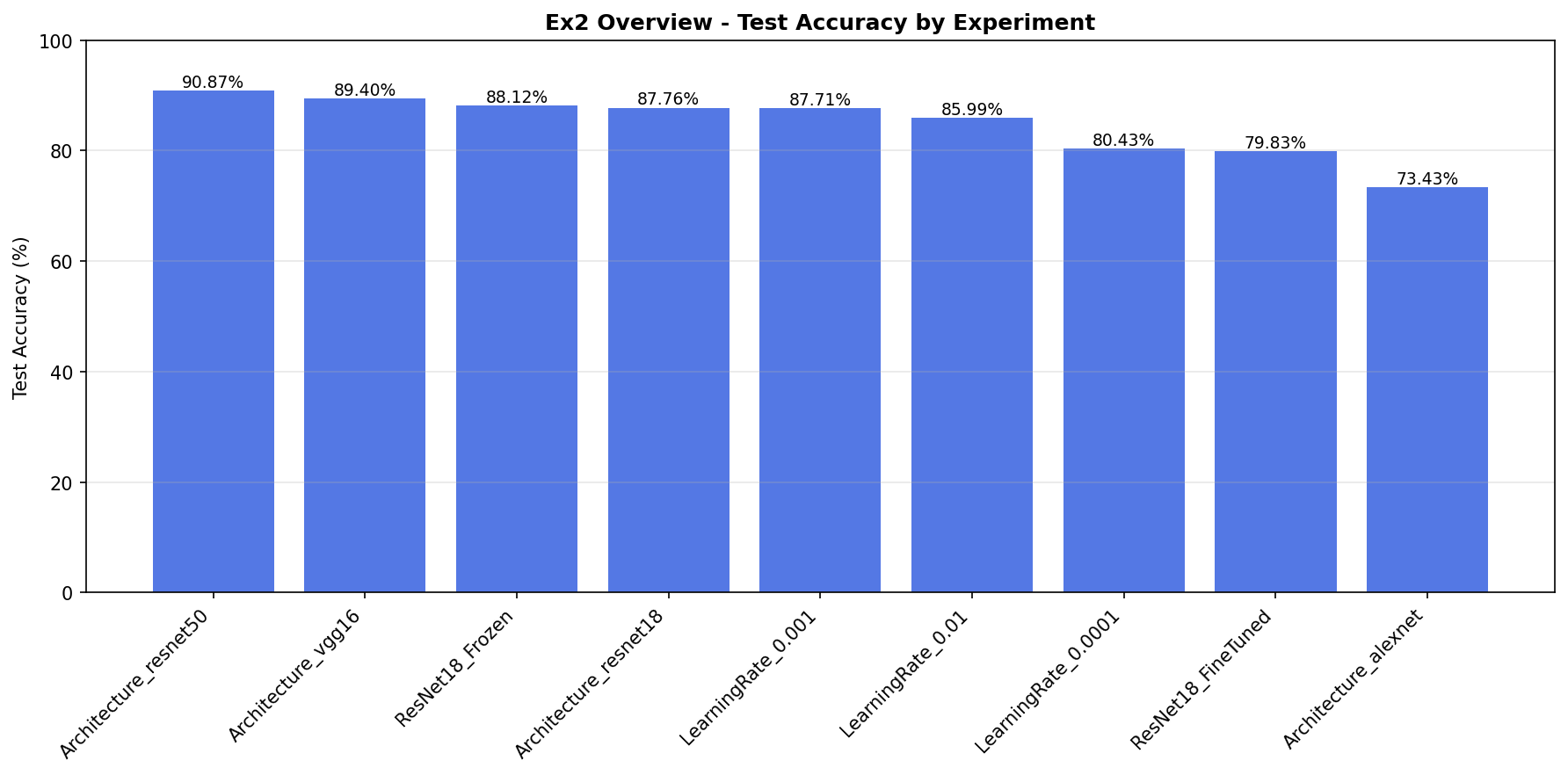
## 2.1 Εισαγωγή

Εφαρμόστηκε transfer learning στο Oxford-IIIT Pet dataset (37 ράτσες σκύλων/γατών). Χρησιμοποιήθηκαν προεκπαιδευμένα μοντέλα ImageNet: τα συνελικτικά στρώματα (feature extractor) παγώνουν και αντικαθίσταται μόνο το classifier head για 37 κλάσεις. 9 πειράματα σε 3 σειρές (20 epochs, step scheduler, Adam):

* Σειρά 1 — Αρχιτεκτονικές: ResNet18, ResNet50, AlexNet, VGG16 (όλες frozen)
* Σειρά 2 — Frozen vs Fine-tuned: ResNet18 με παγωμένα ή ελεύθερα βάρη
* Σειρά 3 — Learning rates: 0.0001, 0.001, 0.01 (ResNet18, frozen)

## 2.2 Συνοπτικός Πίνακας Αποτελεσμάτων

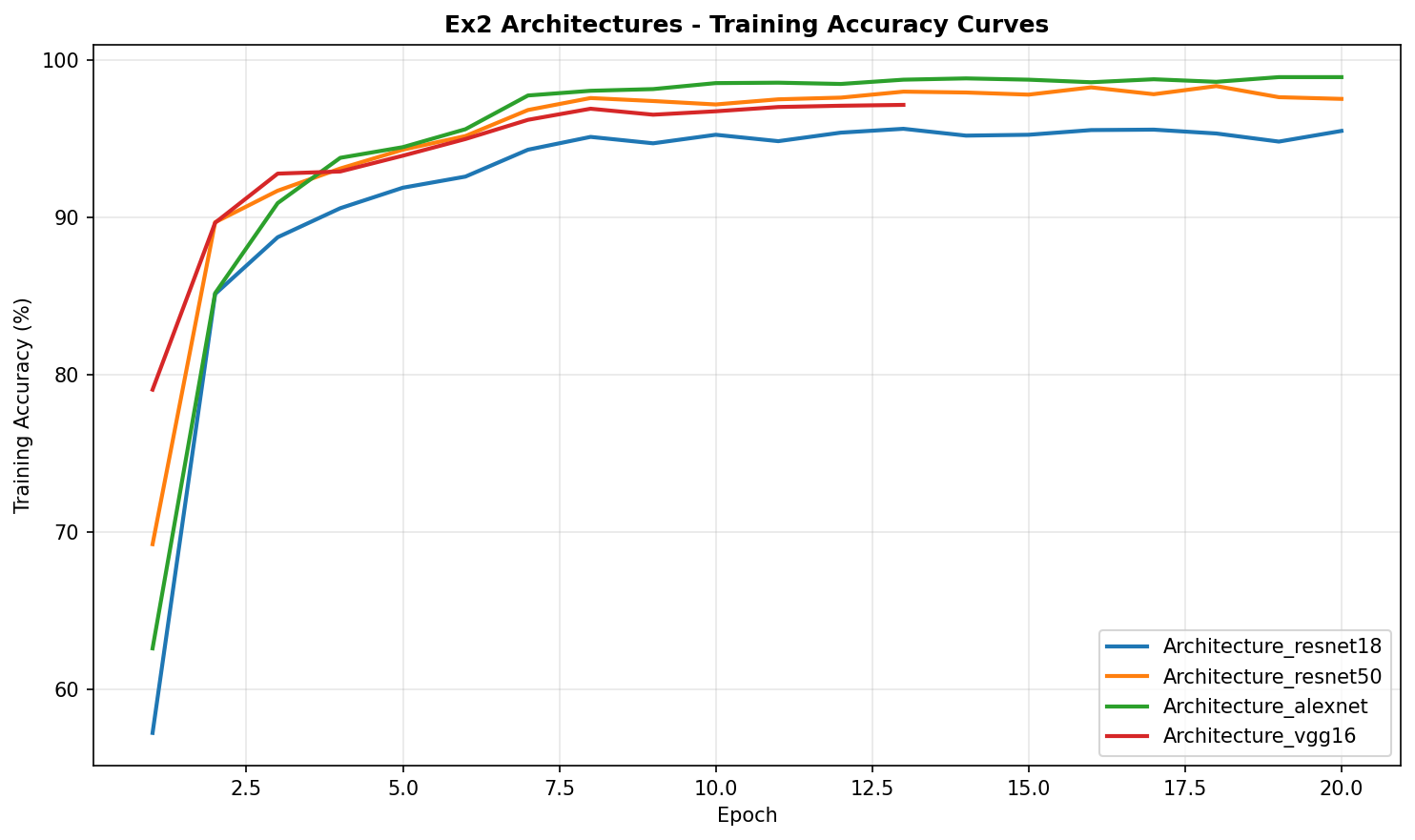
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Πείραμα** | **Model** | **LR** | **Frozen** | **Trainable** | **Train(%)** | **Test(%)** | **Χρόνος(min)** |
| Architecture\_resnet50 | resnet50 | 0.001 | True | 75,813 | 98.4 | 90.87 | 34.9 |
| Architecture\_vgg16 | vgg16 | 0.001 | True | 151,589 | 97.2 | 89.40 | 37.7 |
| ResNet18\_Frozen | resnet18 | 0.001 | True | 18,981 | 95.8 | 88.12 | 10.8 |
| Architecture\_resnet18 | resnet18 | 0.001 | True | 18,981 | 95.7 | 87.76 | 10.5 |
| LearningRate\_0.001 | resnet18 | 0.001 | True | 18,981 | 95.8 | 87.71 | 9.5 |
| LearningRate\_0.01 | resnet18 | 0.01 | True | 18,981 | 98.8 | 85.99 | 9.5 |
| LearningRate\_0.0001 | resnet18 | 0.0001 | True | 18,981 | 84.2 | 80.43 | 9.5 |
| ResNet18\_FineTuned | resnet18 | 0.001 | False | 11,195,493 | 100.0 | 79.83 | 25.7 |
| Architecture\_alexnet | alexnet | 0.001 | True | 151,589 | 98.9 | 73.43 | 3.9 |



*Εικόνα 2.1: Επισκόπηση ακρίβειας όλων των πειραμάτων*

## 2.3 Σύγκριση Αρχιτεκτονικών

Όλες οι αρχιτεκτονικές χρησιμοποιούν pretrained ImageNet βάρη με frozen features. Μόνο το classifier head εκπαιδεύεται:



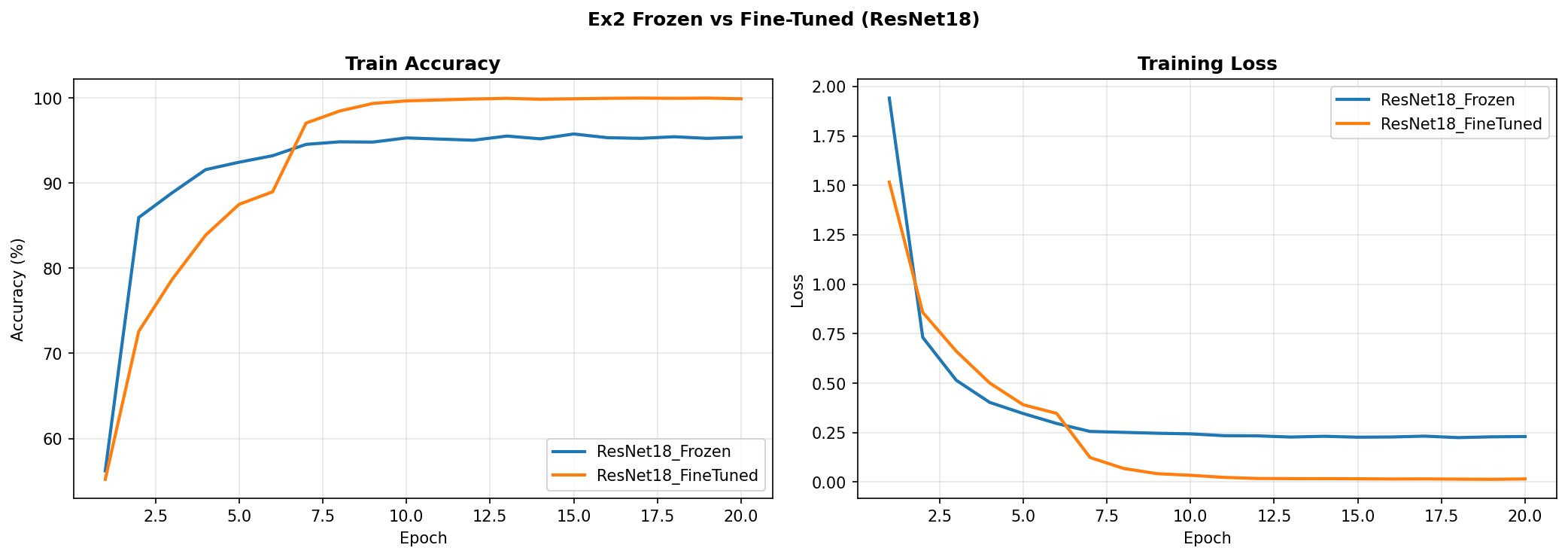
*Εικόνα 2.2: Καμπύλες εκπαίδευσης ανά αρχιτεκτονική*

Ανάλυση:

* ResNet50 (90.87%): Καλύτερο αποτέλεσμα. Το ResNet50 έχει 50 στρώματα και residual connections, εξάγοντας πιο πλούσια και ιεραρχικά features από το ImageNet. Το βάθος επιτρέπει ανίχνευση λεπτών διαφορών μεταξύ ρατσών.
* VGG16 (89.40%): Πολύ κοντά στο ResNet50 παρά τα 134M parameters. Το VGG16 έχει απλή αρχιτεκτονική (3x3 convolutions μόνο) αλλά τα πολλά στρώματα δίνουν ισχυρά features. Όμως χρειάζεται 6x περισσότερο χρόνο (37.7 vs 10.5 min).
* ResNet18 (87.76%): Καλή ισορροπία ακρίβειας/ταχύτητας. Με μόνο 18.981 εκπαιδεύσιμες παραμέτρους, πετυχαίνει 87.76% σε 10.5 λεπτά — μόλις 3% κάτω από το ResNet50.
* AlexNet (73.43%): Χαμηλότερο κατά 17%. Το AlexNet έχει μόνο 5 conv layers με μεγάλα kernels (11x11, 5x5), που χάνουν λεπτομέρειες texture. Παρότι πετυχαίνει 98.94% στο train (overfitting), τα features του δεν μεταφέρονται καλά στο πρόβλημα των ρατσών.

## 2.4 Frozen vs Fine-tuned

Σύγκριση ResNet18 με παγωμένα (frozen) ή ελεύθερα (fine-tuned) βάρη στο feature extractor:



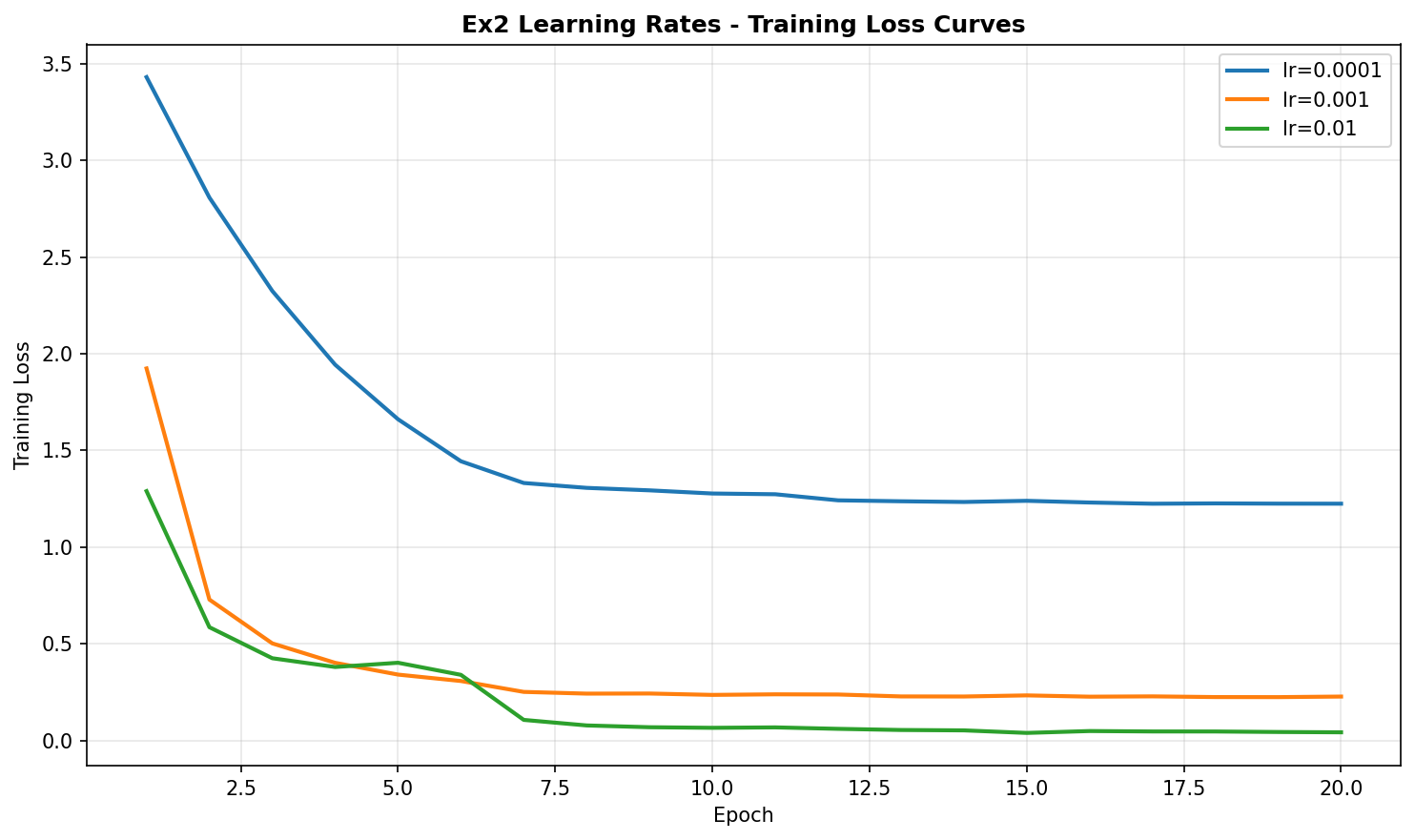
*Εικόνα 2.3: Frozen vs Fine-tuned — Καμπύλες εκπαίδευσης*

Ανάλυση:

* Frozen (88.12%): Εκπαιδεύονται μόνο 18.981 παράμετροι (classifier). Τα ImageNet features είναι ήδη εξαιρετικά γενικά και μεταφέρονται άμεσα. Το μοντέλο δεν κάνει overfit (train 95.76%) και γενικεύει καλά.
* Fine-tuned (79.83%): Εκπαιδεύονται όλες οι 11.2M παράμετροι. Το train accuracy φτάνει 99.97%, αλλά το test πέφτει στο 79.83% — σοβαρό overfitting. Με μόνο ~3.680 εικόνες train, το δίκτυο με 11M παραμέτρους απομνημονεύει τα train data αντί να γενικεύει.
* Συμπέρασμα: Με μικρό dataset, το freezing είναι καλύτερο γιατί λειτουργεί ως regularization. Το fine-tuning χρειάζεται πολύ χαμηλό lr, data augmentation, ή dropout για να αποφευχθεί το overfitting.

## 2.5 Επίδραση Learning Rate

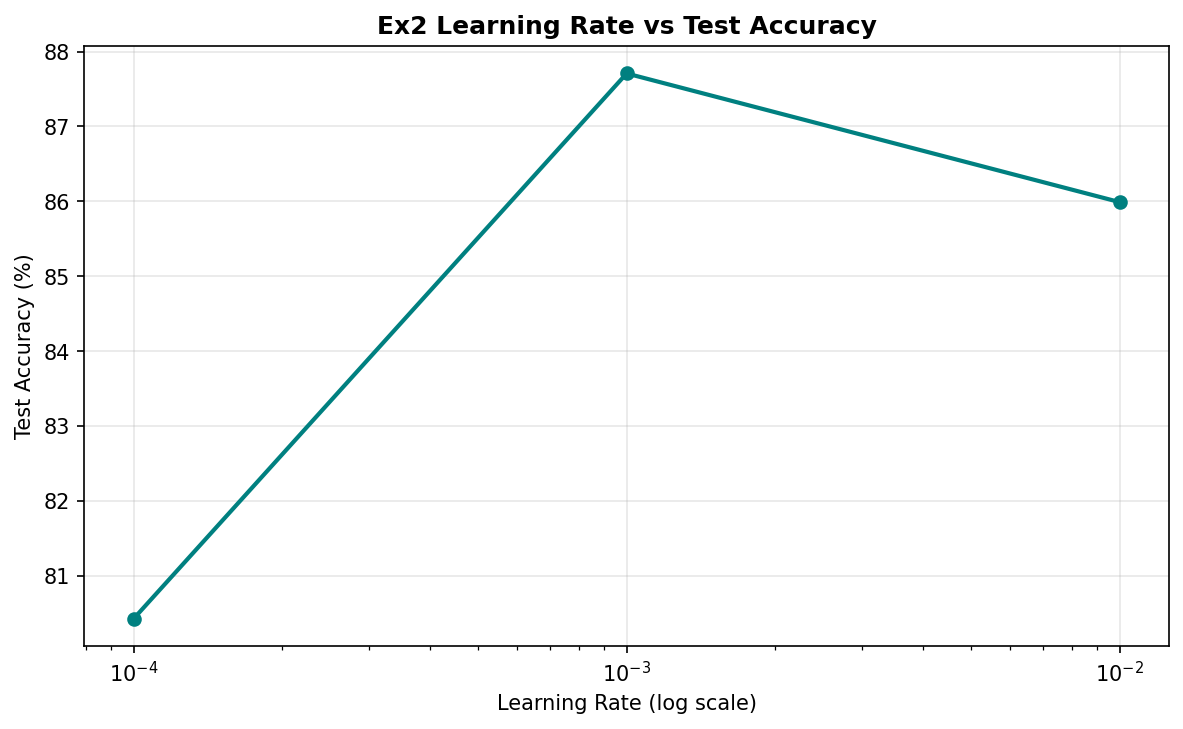
Σύγκριση τριών learning rates με ResNet18 frozen:



*Εικόνα 2.4: Καμπύλες training loss ανά learning rate*

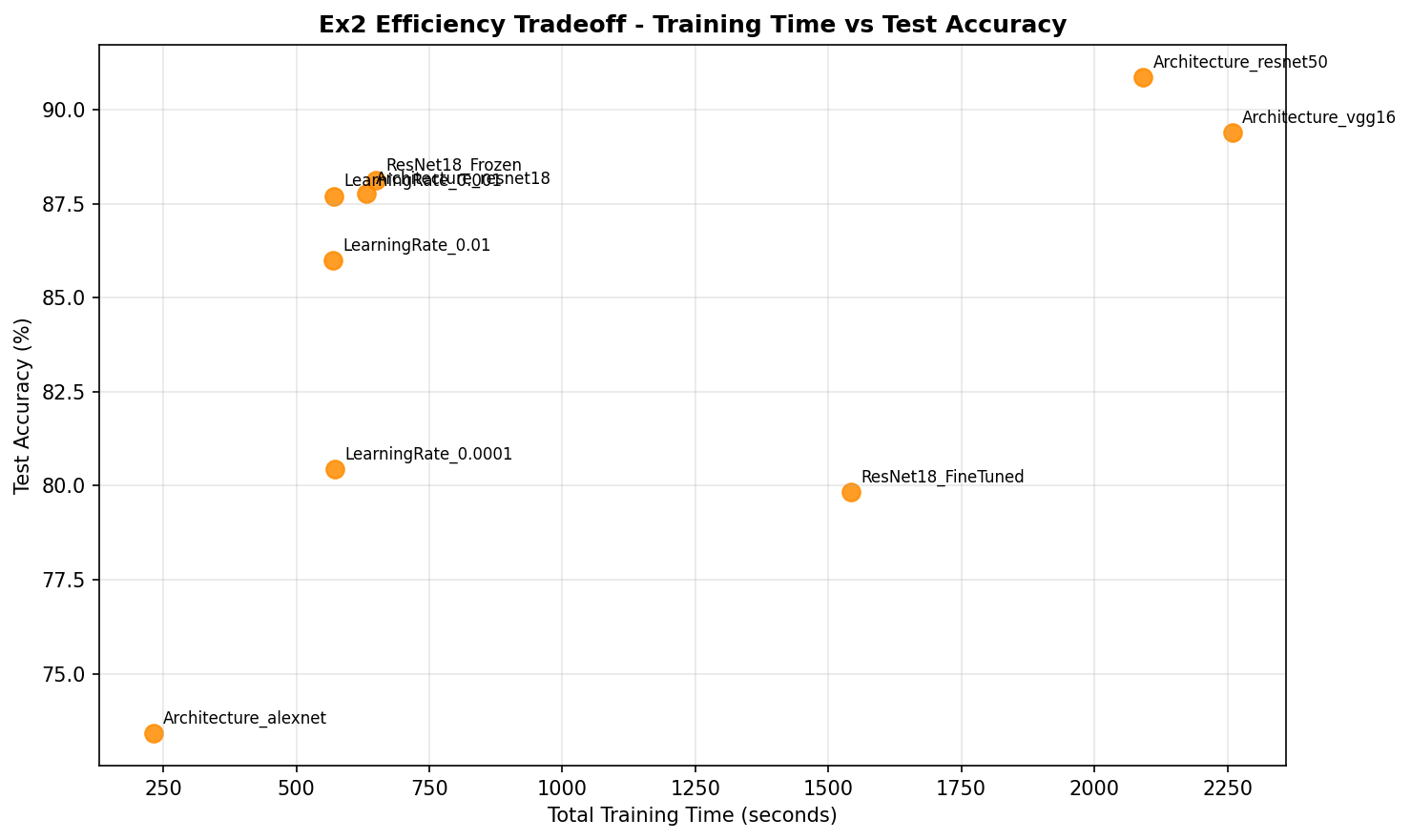
Ανάλυση:

* lr=0.001 (87.71%): Ιδανικό. Ο Adam λειτουργεί άριστα με αυτό το lr στο transfer learning, γιατί είναι αρκετά υψηλό για γρήγορη σύγκλιση, αλλά όχι τόσο υψηλό ώστε να προκαλέσει overfit στο classifier.
* lr=0.01 (85.99%): Ελαφρύ overfitting. Το μοντέλο φτάνει 98.83% train αλλά 86% test. Το υψηλό lr κάνει το classifier να προσαρμόζεται υπερβολικά στα train data. Παρατηρείται σημαντική πτώση loss στο epoch 7 (λόγω step scheduler).
* lr=0.0001 (80.43%): Πολύ αργή σύγκλιση. Σε 20 epochs το train accuracy έφτασε μόλις 84.24%. Το loss δεν κατέβηκε αρκετά (1.22 vs 0.23 με lr=0.001). Θα χρειαζόταν 60+ epochs για να φτάσει το επίπεδο του lr=0.001.



*Εικόνα 2.5: Learning rate vs test accuracy*

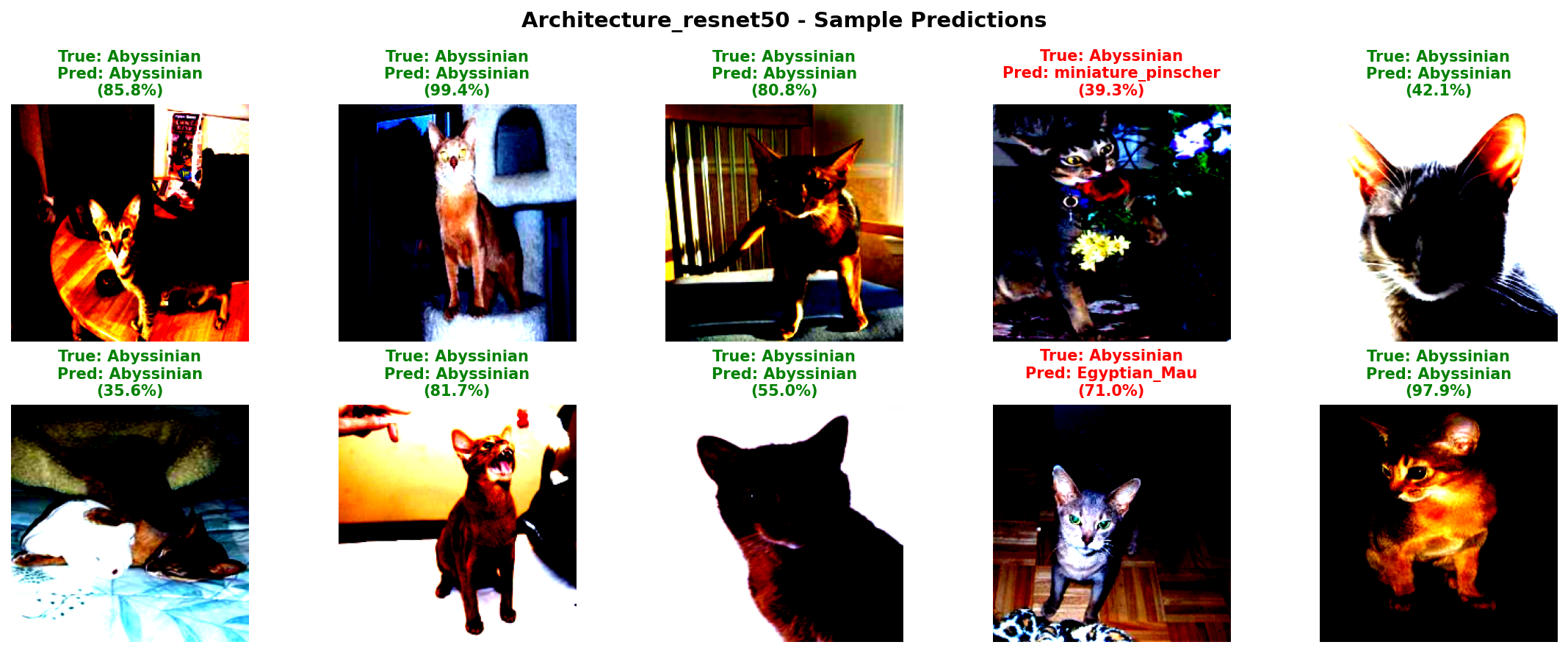
## 2.6 Αποδοτικότητα Χρόνου



*Εικόνα 2.6: Χρόνος εκπαίδευσης vs test accuracy*

Ο χρόνος διαφέρει έντονα: AlexNet (3.9 min) vs VGG16 (37.7 min). Το ResNet18 frozen προσφέρει την καλύτερη ισορροπία ακρίβειας/χρόνου (87.76% σε 10.5 min), ενώ το fine-tuning διπλασιάζει τον χρόνο (25.7 min) χωρίς όφελος στην ακρίβεια.

## 2.7 Ενδεικτικά Αποτελέσματα Ταξινόμησης



*Εικόνα 2.7: Δείγματα προβλέψεων — Καλύτερο μοντέλο (ResNet50, frozen)*

## 2.8 Συμπεράσματα Άσκησης 2

* Το transfer learning πετυχαίνει 90.87% με μόνο 75.813 trainable parameters (ResNet50 frozen) — από 56.43% με 964K parameters στο vanilla CNN (Άσκηση 1).
* Οι βαθύτερες αρχιτεκτονικές εξάγουν καλύτερα features: ResNet50 > VGG16 > ResNet18 > AlexNet.
* Το freezing υπερτερεί του fine-tuning σε μικρά datasets, λειτουργώντας ως ισχυρό regularizer.
* Το lr=0.001 είναι ιδανικό για Adam+frozen features. Το lr αφορά μόνο το classifier, άρα χρειάζεται αρκετή ταχύτητα χωρίς να προκαλεί overfitting.

# Άσκηση #3: Σημασιολογική Τμηματοποίηση (Semantic Segmentation)

## 3.1 Εισαγωγή

Υλοποιήθηκε U-Net (31M παράμετροι, base channels=64) για semantic segmentation στο SBD dataset (21 κλάσεις Pascal VOC). Το U-Net χρησιμοποιεί encoder-decoder με skip connections για να διατηρεί χωρικές λεπτομέρειες. 7 πειράματα σε 2 σειρές:

* Σειρά 1 — Κανονικό BG weight=1.0: 3 πειράματα, 10 epochs, step scheduler, CrossEntropy
* Σειρά 2 — Μειωμένο BG weight=0.2: 4 πειράματα, 5 epochs, cosine scheduler, WeightedCE

## 3.2 Ρόλος Background Weight

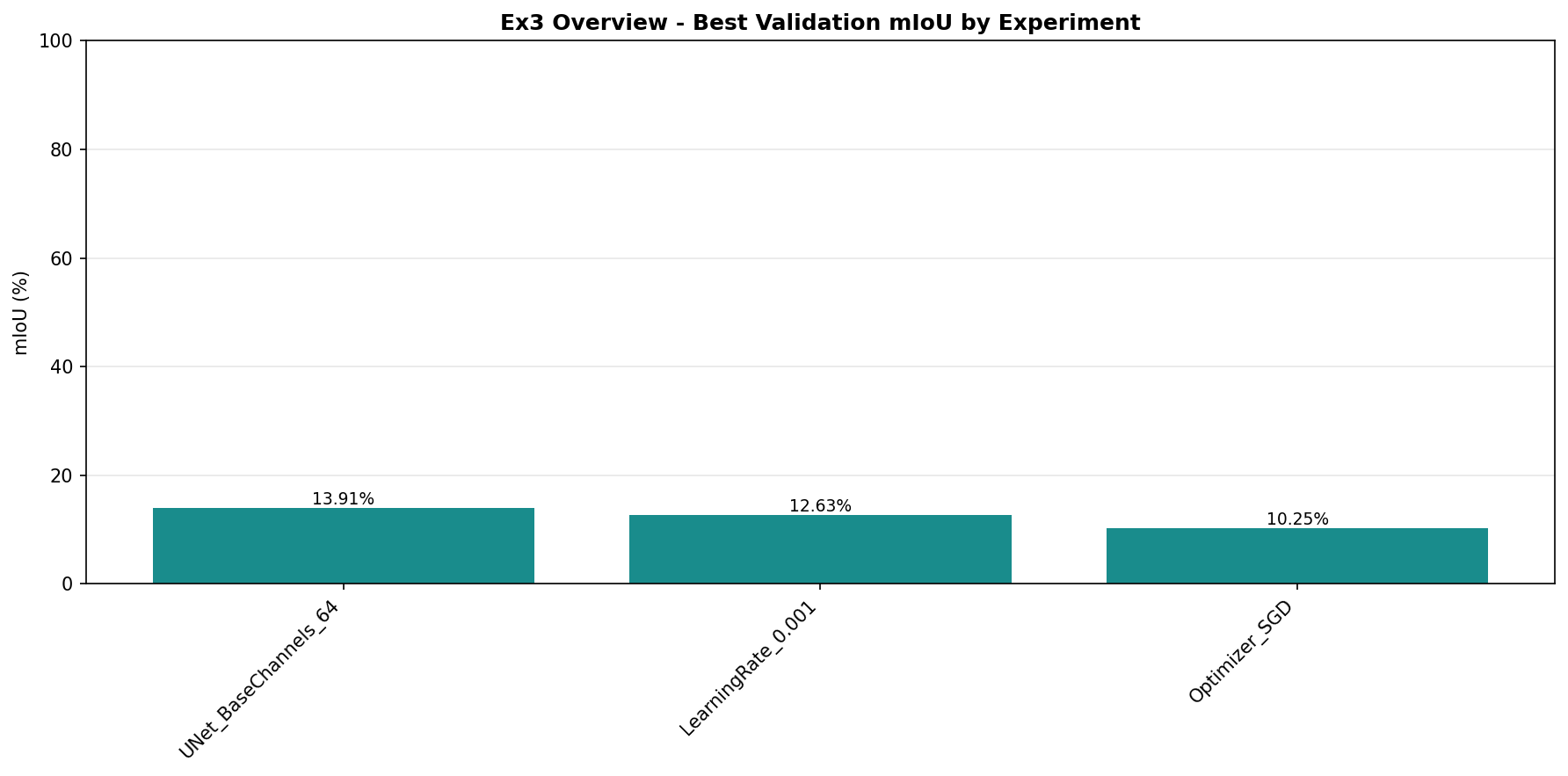
Στο SBD, το background καταλαμβάνει >80% των pixels σε κάθε εικόνα. Με weight=1.0, το μοντέλο βελτιστοποιεί κυρίως για το background και πετυχαίνει υψηλό Pixel Accuracy αλλά χαμηλό mIoU στις foreground κλάσεις. Μειώνοντας το στο 0.2, η loss δίνει περισσότερο βάρος στις foreground κλάσεις, αλλά η εκπαίδευση γίνεται πιο ασταθής και χρειάζεται περισσότερα epochs.

## 3.3 Συνοπτικός Πίνακας Αποτελεσμάτων

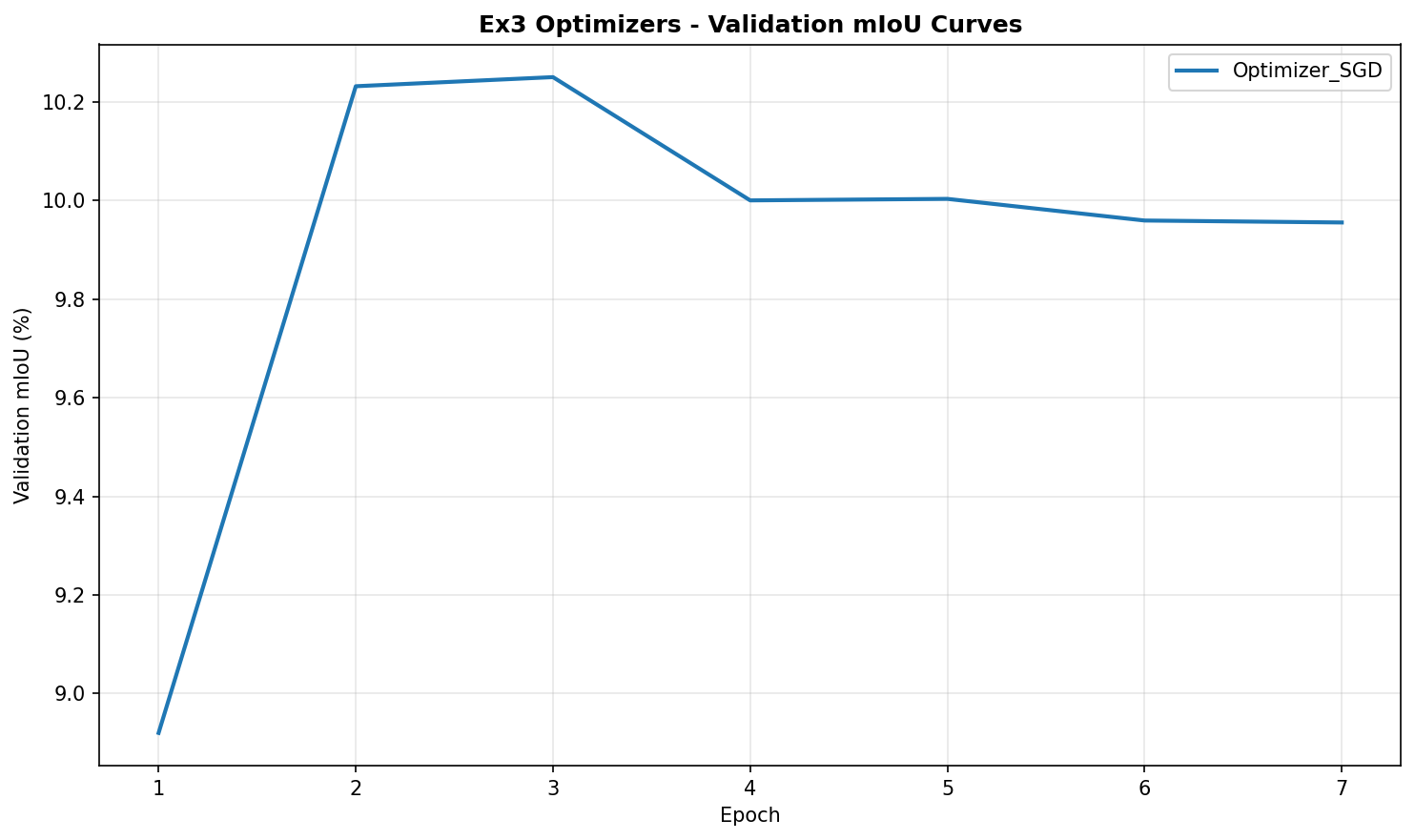
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Πείραμα** | **BG Wt** | **Optimizer** | **LR** | **Epochs** | **mIoU(%)** | **PixAcc(%)** | **Χρόνος(min)** |
| UNet\_BaseChannels\_64 | 1.0 | ADAMW | 0.0003 | 10 | 13.91 | 74.79 | 81.7 |
| LearningRate\_0.001 | 1.0 | ADAMW | 0.001 | 10 | 12.63 | 73.67 | 81.6 |
| Optimizer\_SGD | 1.0 | SGD | 0.0003 | 7 | 10.25 | 71.46 | 56.8 |
| LearningRate\_0.0001 | 0.2 | ADAMW | 0.0001 | 5 | 7.56 | 71.06 | 50.0 |
| Optimizer\_ADAMW | 0.2 | ADAMW | 0.0003 | 5 | 6.75 | 70.22 | 49.6 |
| LearningRate\_0.001 | 0.2 | ADAMW | 0.001 | 5 | 5.80 | 68.42 | 49.8 |
| Optimizer\_SGD | 0.2 | SGD | 0.0003 | 5 | 3.02 | 67.10 | 49.8 |

## 3.4 Σειρά 1: Κανονικό BG Weight (=1.0)

3 πειράματα με 10 epochs, step scheduler. Σύγκριση AdamW vs SGD και διαφορετικών learning rates:



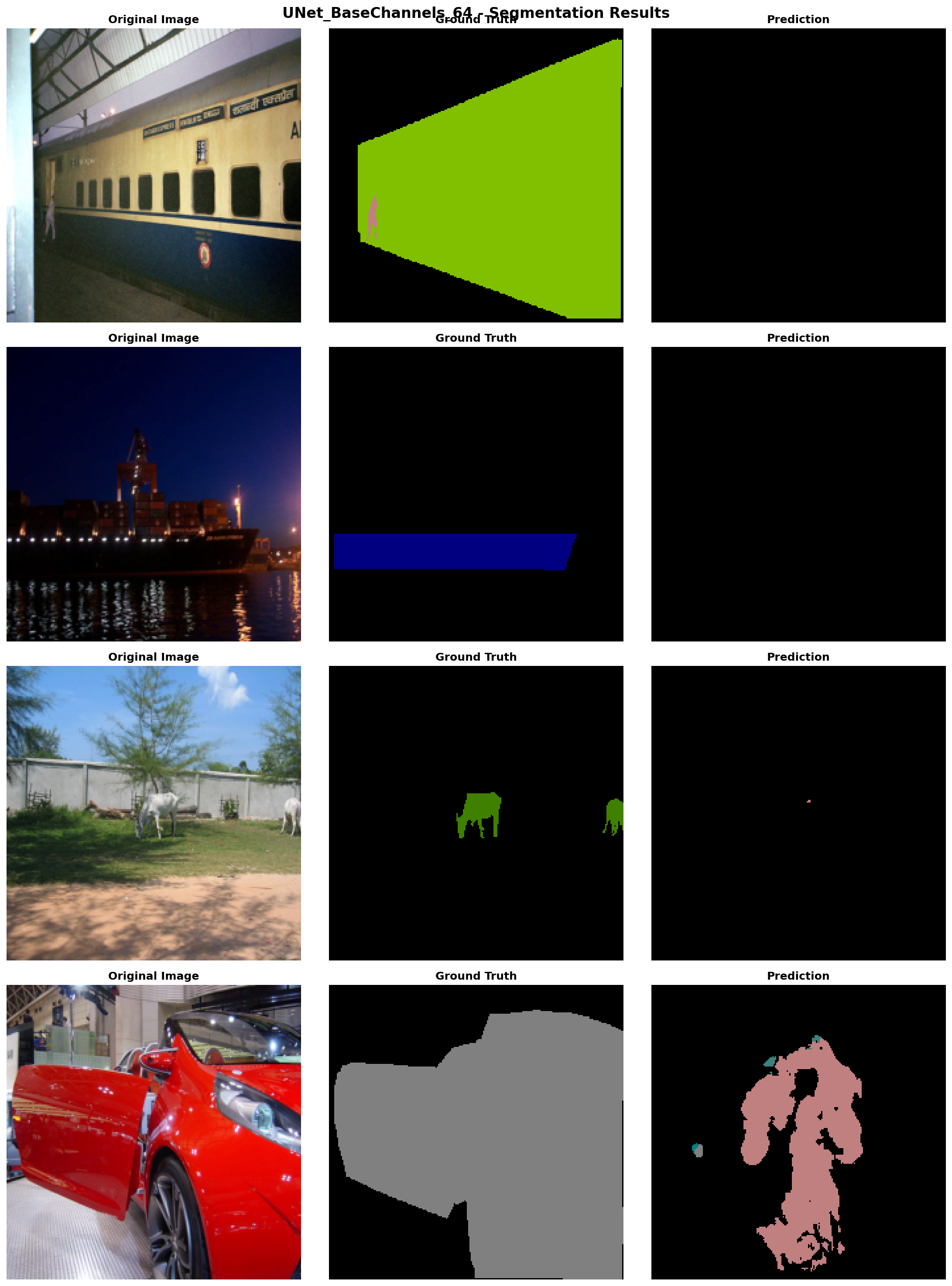
*Εικόνα 3.1: Σύγκριση mIoU — Κανονικό BG Weight*



*Εικόνα 3.2: Καμπύλες mIoU ανά πείραμα (BG=1.0)*

Ανάλυση:

* BaseChannels\_64 (AdamW, lr=0.0003): Καλύτερο με mIoU=13.91%, PixAcc=74.79%. Ο AdamW συνδυάζει adaptive LR με weight decay, προσφέροντας σταθερή σύγκλιση σε ένα δίκτυο με 31M παραμέτρους.
* Optimizer\_SGD (lr=0.0003): Χαμηλότερο mIoU=10.19%, PixAcc=71.11%. Ο SGD χωρίς adaptive scaling συγκλίνει πιο αργά σε μεγάλα δίκτυα. Σε αντίθεση με την Άσκηση 1 (όπου ο SGD ήταν καλύτερος), εδώ το μέγεθος του U-Net απαιτεί adaptive optimizer.
* LearningRate\_0.001 (AdamW): mIoU=12.88%, PixAcc=73.65%. Το υψηλότερο LR προκαλεί ελαφρώς χαμηλότερο mIoU — το 0.0003 είναι πιο κατάλληλο για fine-grained segmentation όπου η ακρίβεια pixel-level είναι κρίσιμη.



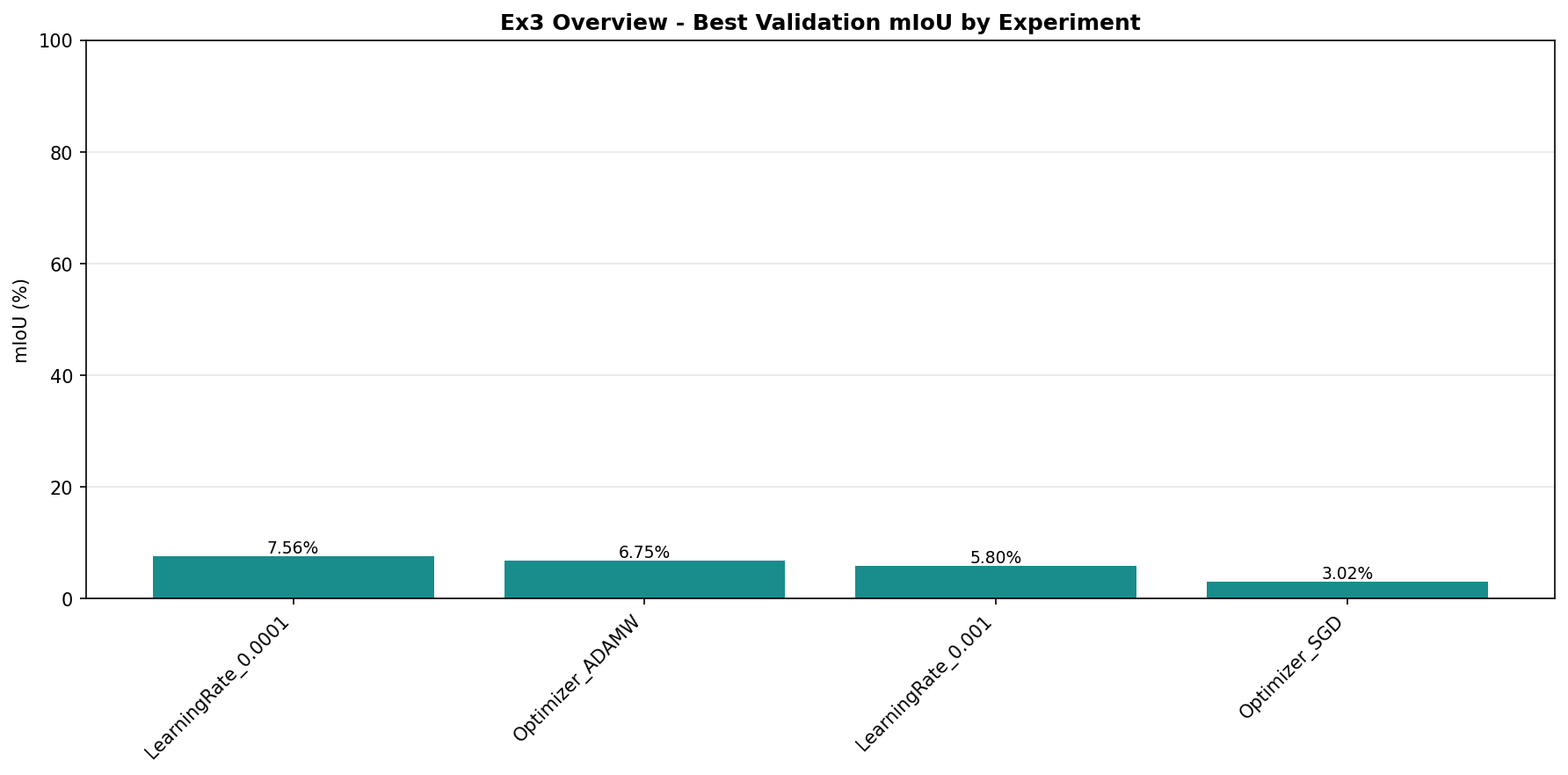
*Εικόνα 3.3: Καλύτερο αποτέλεσμα τμηματοποίησης (BG=1.0, AdamW)*



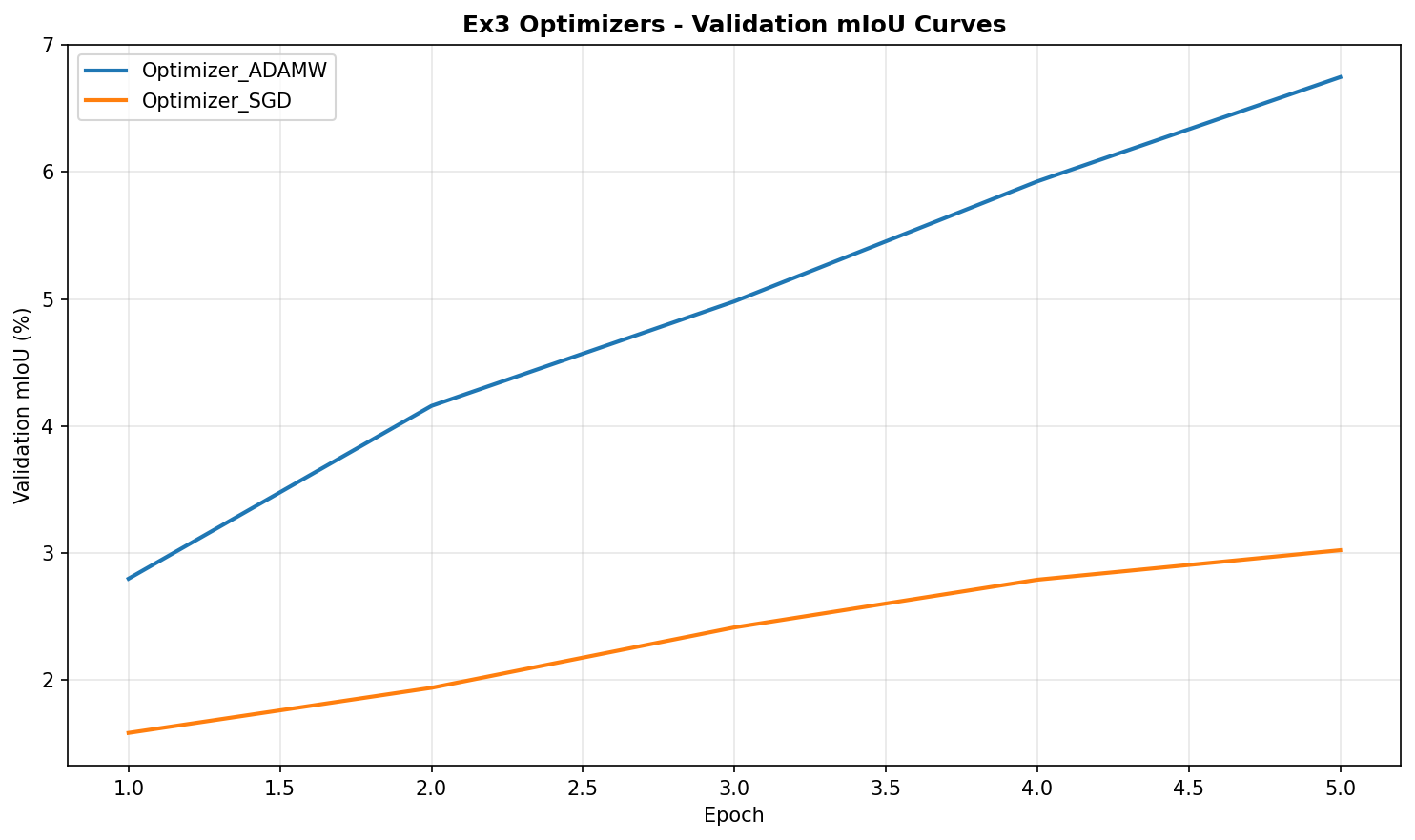
*Εικόνα 3.4: Χρόνος vs mIoU (BG=1.0)*

## 3.5 Σειρά 2: Μειωμένο BG Weight (=0.2)

4 πειράματα με 5 epochs, cosine scheduler. Στόχος: να δούμε αν η μείωση του BG weight βελτιώνει το foreground mIoU:



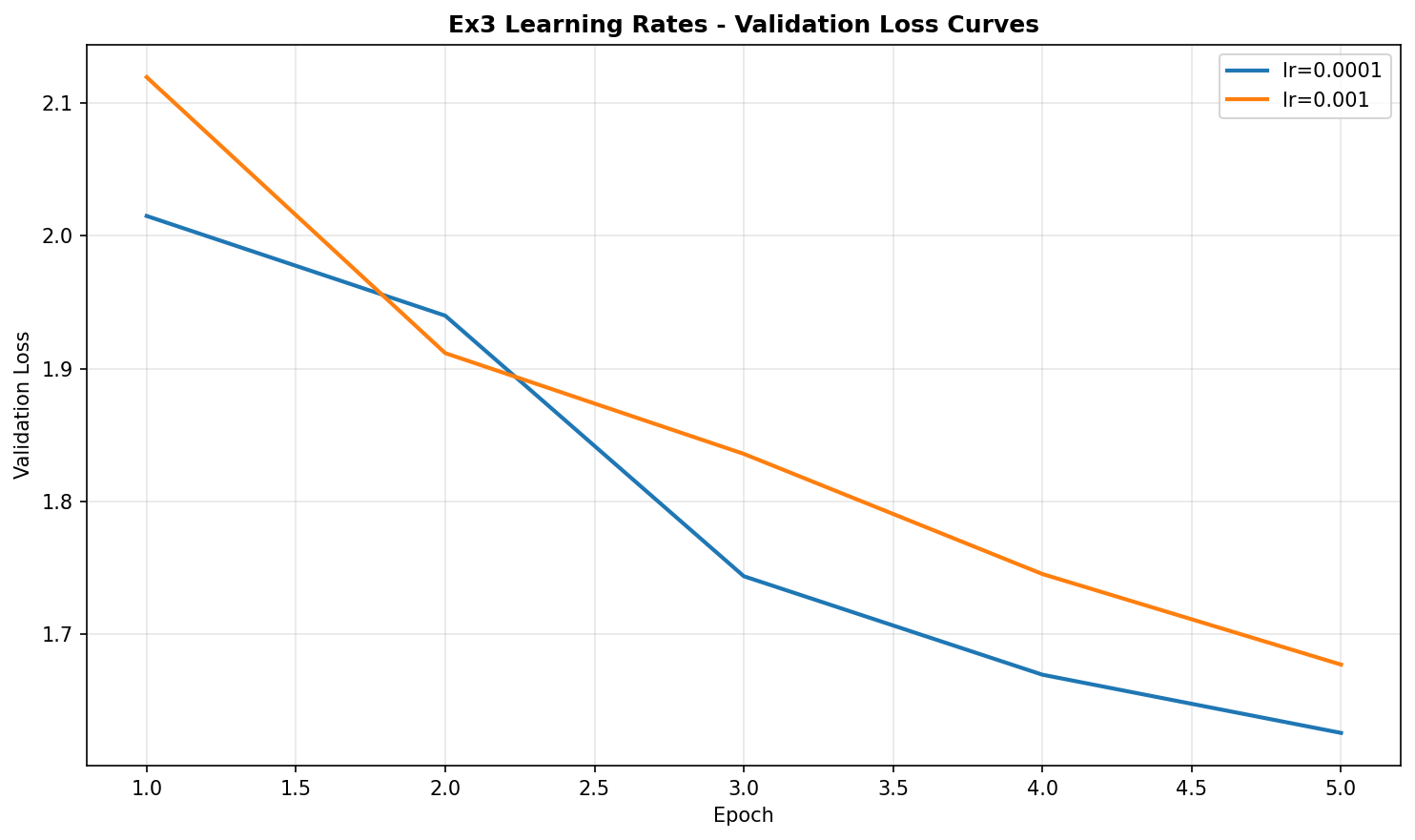
*Εικόνα 3.5: Σύγκριση mIoU — Μειωμένο BG Weight (0.2)*



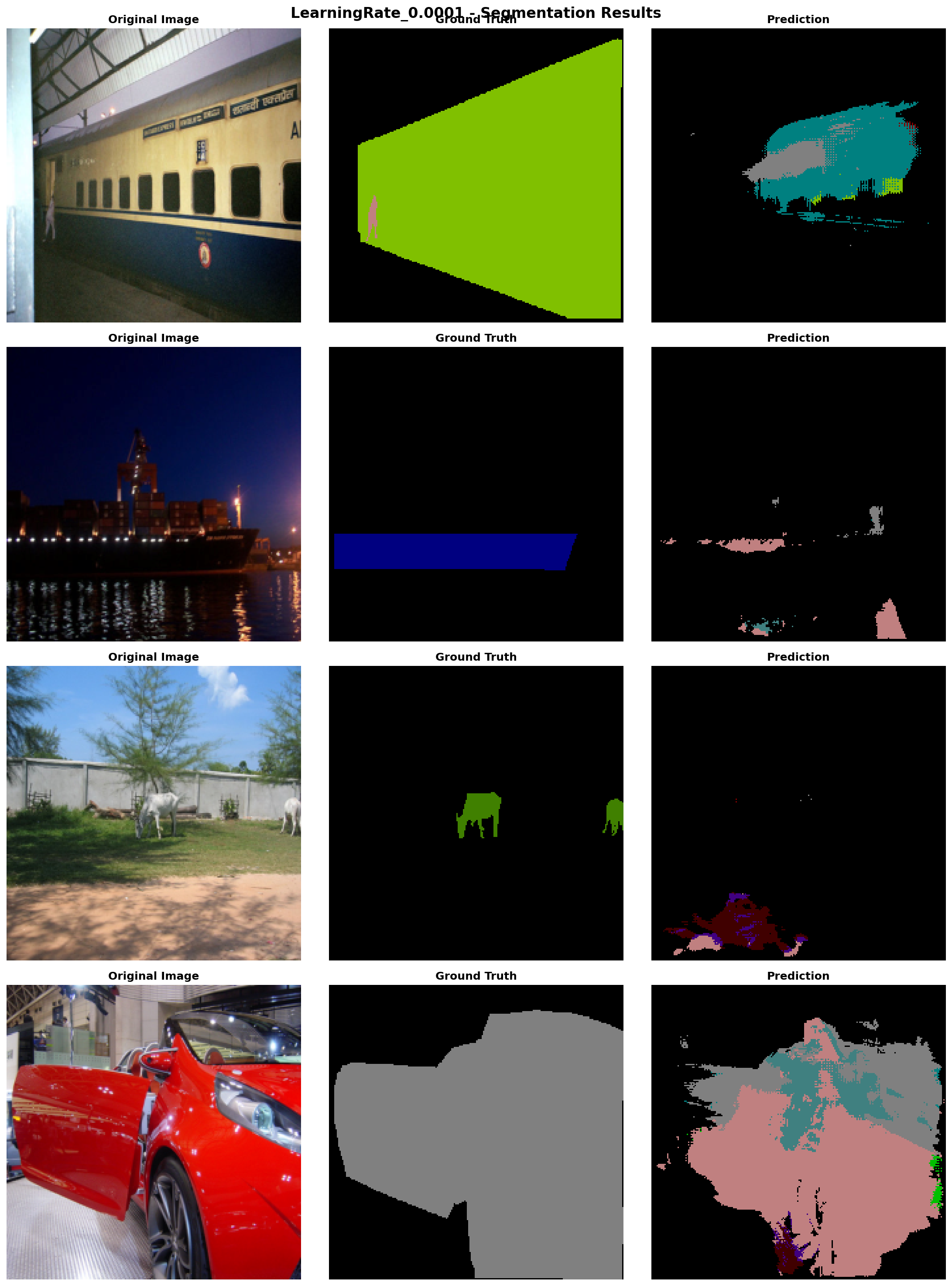
*Εικόνα 3.6: Καμπύλες mIoU ανά πείραμα (BG=0.2)*

Ανάλυση:

* LR\_0.0001 + AdamW (mIoU=7.56%): Καλύτερο της σειράς 2. Το πολύ χαμηλό lr σταθεροποιεί την εκπαίδευση παρά τα ανισόρροπα class weights. Με περισσότερα epochs αναμένεται σημαντική βελτίωση.
* LR\_0.001 + AdamW (mIoU=5.55%): Το υψηλότερο LR σε συνδυασμό με BG=0.2 προκαλεί αστάθεια: το μοντέλο υπερ-αντιδρά στα μειωμένα background gradients και δεν συγκλίνει σωστά.
* SGD (mIoU=3.04-5.07%): Ακόμα χαμηλότερα. Ο SGD δεν μπορεί να προσαρμοστεί στα ανισόρροπα gradients που προκαλεί το BG=0.2 σε μόλις 5 epochs.



*Εικόνα 3.7: Validation loss ανά LR (BG=0.2)*



*Εικόνα 3.8: Καλύτερο αποτέλεσμα τμηματοποίησης (BG=0.2)*

## 3.6 Σύγκριση BG=1.0 vs BG=0.2

Ανάλυση:

* BG=1.0: Σταθερή εκπαίδευση και υψηλότερο mIoU (10-14%) και PixAcc (71-75%). Το μοντέλο μαθαίνει καλά το background και σταδιακά βελτιώνει και τις foreground κλάσεις.
* BG=0.2: Χαμηλότερα mIoU (3-7.5%) και PixAcc (13-55%). Η κύρια αιτία είναι ότι 5 epochs δεν είναι αρκετά — το μοντέλο δεν προλαβαίνει να συγκλίνει. Επίσης, το μειωμένο BG weight προκαλεί μεγαλύτερη gradient variance.
* Με ίσα epochs (10), το μειωμένο BG weight αναμένεται να υπερτερήσει στο foreground mIoU, γιατί θα δώσει περισσότερη προσοχή στις σπάνιες κλάσεις (person, car, dog κλπ).

## 3.7 Συμπεράσματα Άσκησης 3

* Ο AdamW είναι ο καταλληλότερος optimizer για U-Net (31M params), σε αντίθεση με τον SGD που υπερτερούσε στο vanilla CNN (964K params).
* Το χαμηλό lr (0.0003) είναι κρίσιμο για segmentation: το pixel-level πρόβλημα απαιτεί πολύ ακριβή σύγκλιση.
* Το BG weight είναι κρίσιμη υπερ-παράμετρος: η μείωση από 1.0 σε 0.2 αλλάζει ριζικά τη συμπεριφορά του μοντέλου και απαιτεί αντίστοιχη προσαρμογή LR και epochs.
* Το mIoU=13.91% είναι χαμηλό σε σύγκριση με SOTA (π.χ. DeepLabV3+ >70%), αλλά αναμενόμενο για U-Net χωρίς pretrained encoder σε 10 epochs.

# Άσκηση #4: Ανίχνευση Αντικειμένων (Object Detection)

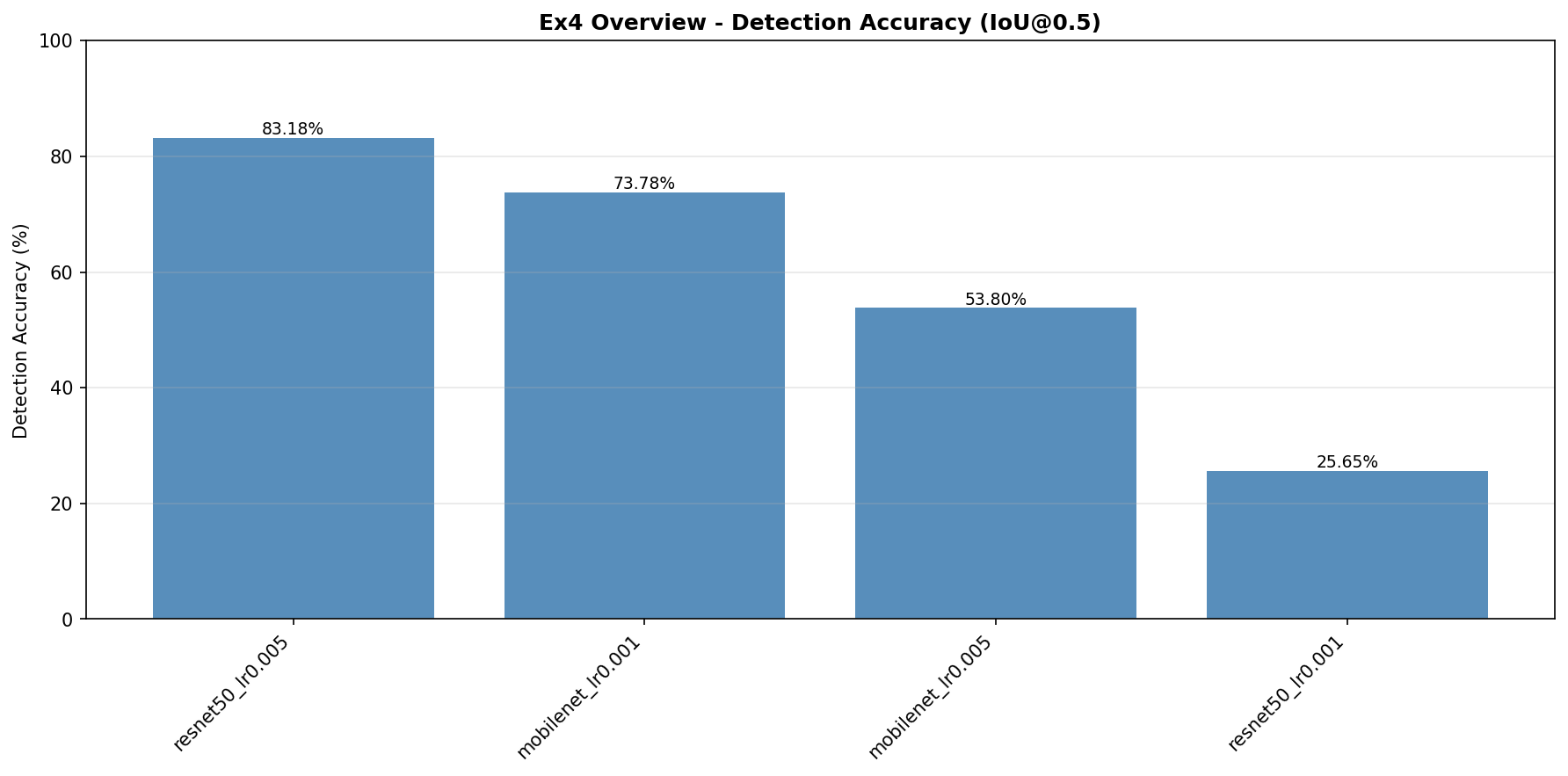
## 4.1 Εισαγωγή

Εφαρμόστηκε Faster R-CNN στο Oxford-IIIT Pet dataset για εντοπισμό ζώων (bounding box + κλάση). Το Faster R-CNN χρησιμοποιεί Region Proposal Network (RPN) και pretrained backbone από το COCO dataset. 4 πειράματα σε 2 άξονες (4 epochs, SGD, step scheduler):

* Άξονας 1 — Backbone: ResNet50-FPN vs MobileNetV3 (βάθος vs ταχύτητα)
* Άξονας 2 — Learning rate: 0.001 vs 0.005

## 4.2 Συνοπτικός Πίνακας Αποτελεσμάτων

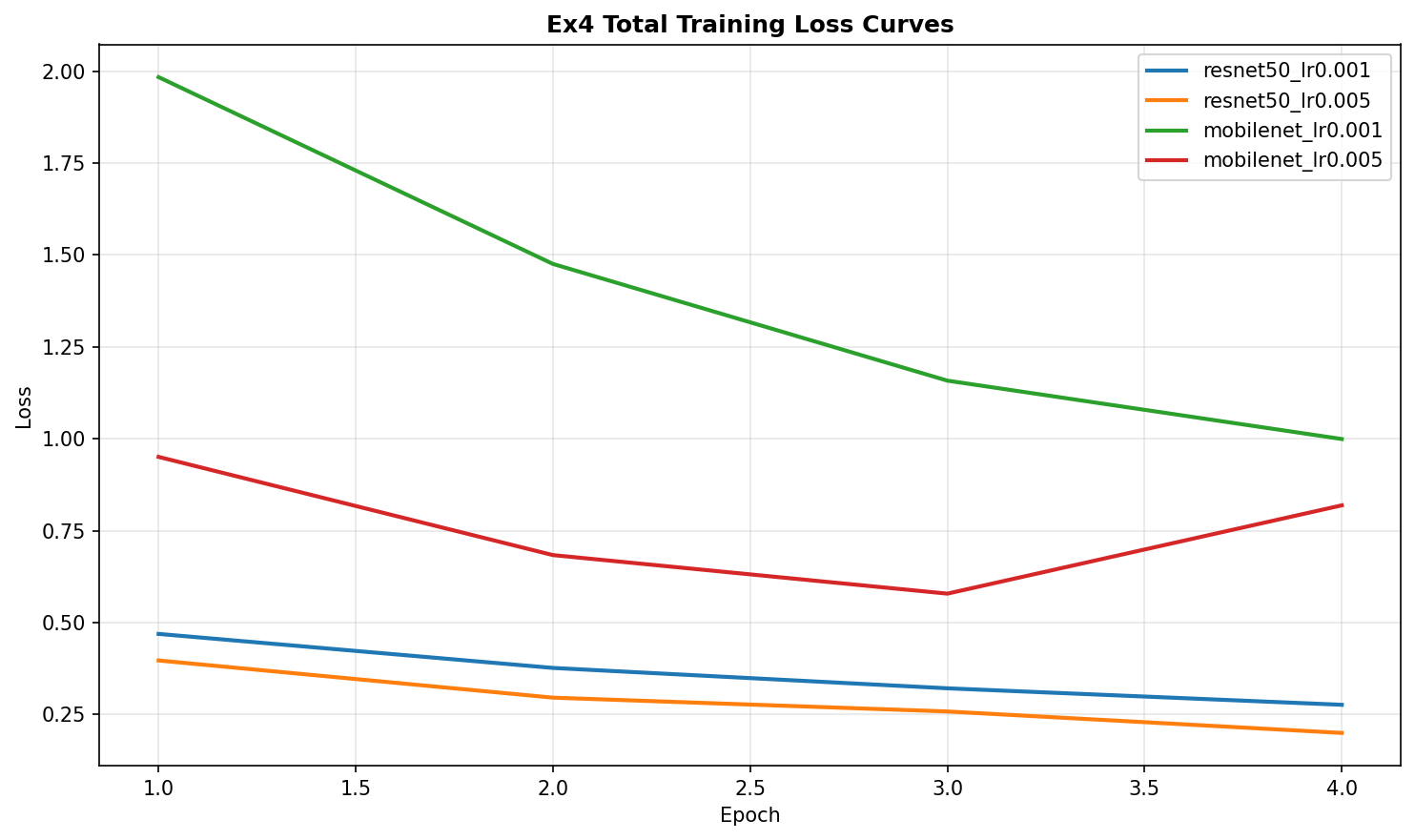
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Πείραμα** | **Backbone** | **LR** | **Det Acc(%)** | **Train Loss** | **Cls Loss** | **Box Loss** |
| resnet50\_lr0.005 | RESNET50 | 0.005 | 83.18 | 0.1998 | 0.0695 | 0.0648 |
| mobilenet\_lr0.001 | MOBILENET | 0.001 | 73.78 | 0.9994 | 0.4762 | 0.4100 |
| mobilenet\_lr0.005 | MOBILENET | 0.005 | 53.80 | 0.8190 | 0.3900 | 0.3250 |
| resnet50\_lr0.001 | RESNET50 | 0.001 | 25.65 | 0.2763 | 0.1273 | 0.0791 |



*Εικόνα 4.1: Επισκόπηση ακρίβειας ανίχνευσης*

## 4.3 Σύγκριση Backbones

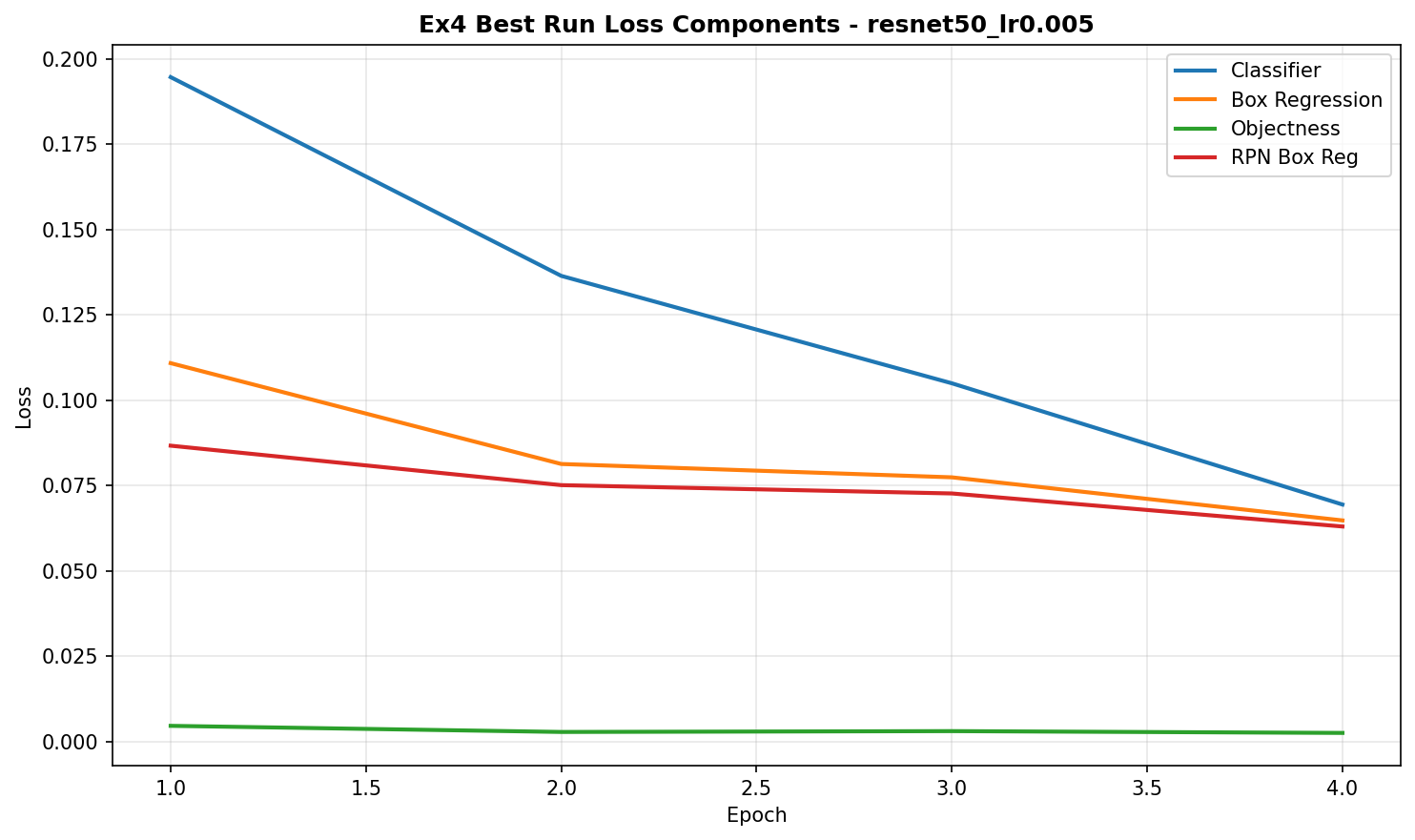
Σύγκριση ResNet50-FPN vs MobileNetV3 ως backbone του Faster R-CNN:



*Εικόνα 4.2: Καμπύλες total loss ανά πείραμα*

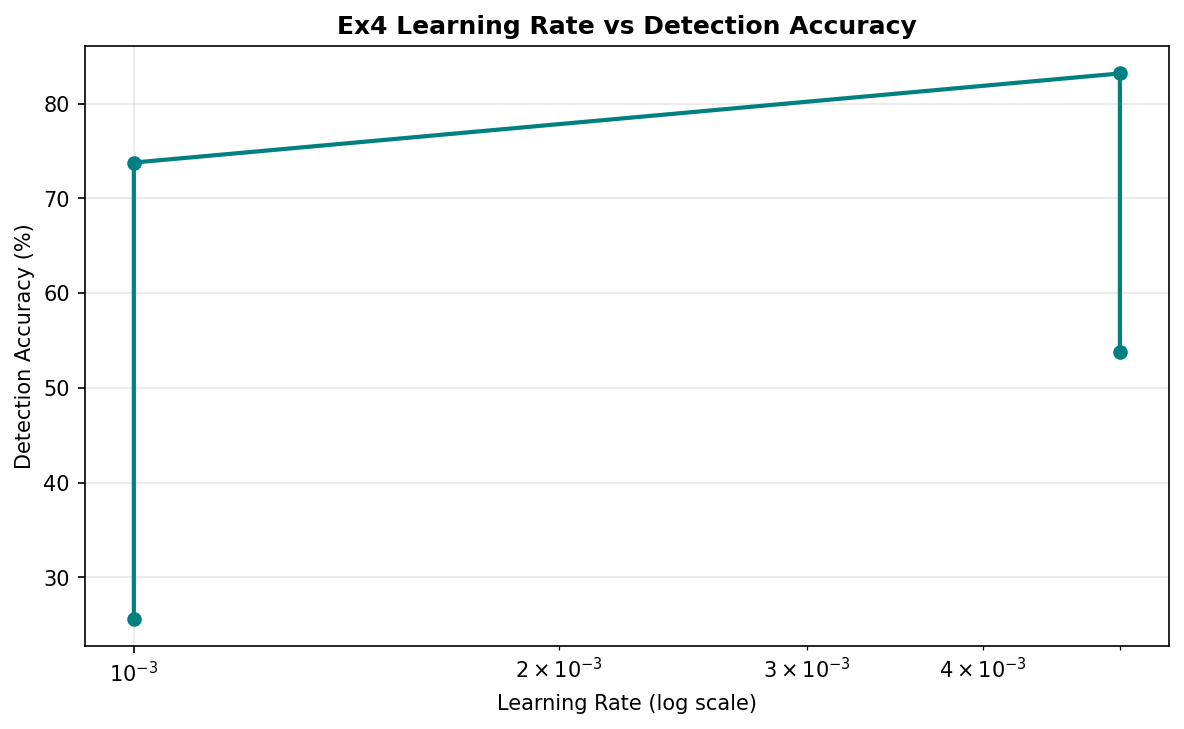
Ανάλυση:

* ResNet50 + lr=0.005 (83.18%): Καλύτερο. Το ResNet50-FPN παρέχει πλούσια multi-scale features μέσω Feature Pyramid Network. Με lr=0.005 το loss έπεσε στο 0.1998, επιτρέποντας ακριβής localization και classification.
* ResNet50 + lr=0.001 (25.65%): Αποτυχία παρά το ίδιο backbone. Το χαμηλό lr δεν επαρκεί για να προσαρμόσει το RPN και τον detection head σε 4 epochs. Το loss έμεινε στο 0.276, πολύ υψηλότερο από το lr=0.005.
* MobileNet + lr=0.001 (73.78%): Καλύτερο από το ResNet50 με ίδιο lr! Το MobileNetV3 είναι ελαφρύτερο και συγκλίνει γρηγορότερα με χαμηλό lr. Αυτό δείχνει ότι το ιδανικό lr εξαρτάται από το backbone.
* MobileNet + lr=0.005 (53.80%): Το υψηλό lr προκαλεί αστάθεια στο ελαφρύ backbone (loss αυξήθηκε από 0.578 σε 0.819). Το MobileNet με λιγότερες παραμέτρους είναι πιο ευαίσθητο σε υψηλό lr.

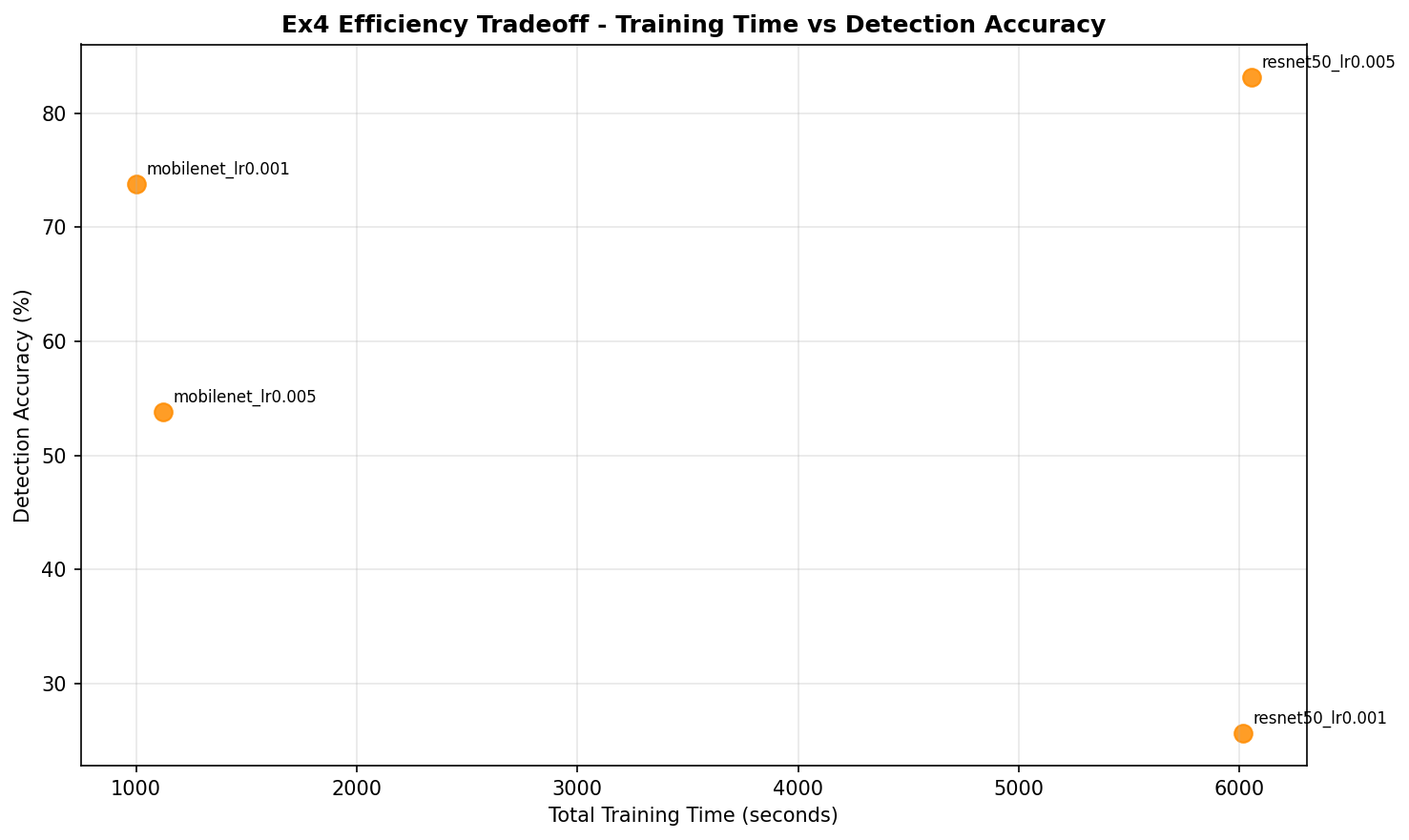


*Εικόνα 4.3: Ανάλυση loss components (classifier + box regression)*

## 4.4 Learning Rate και Χρόνος



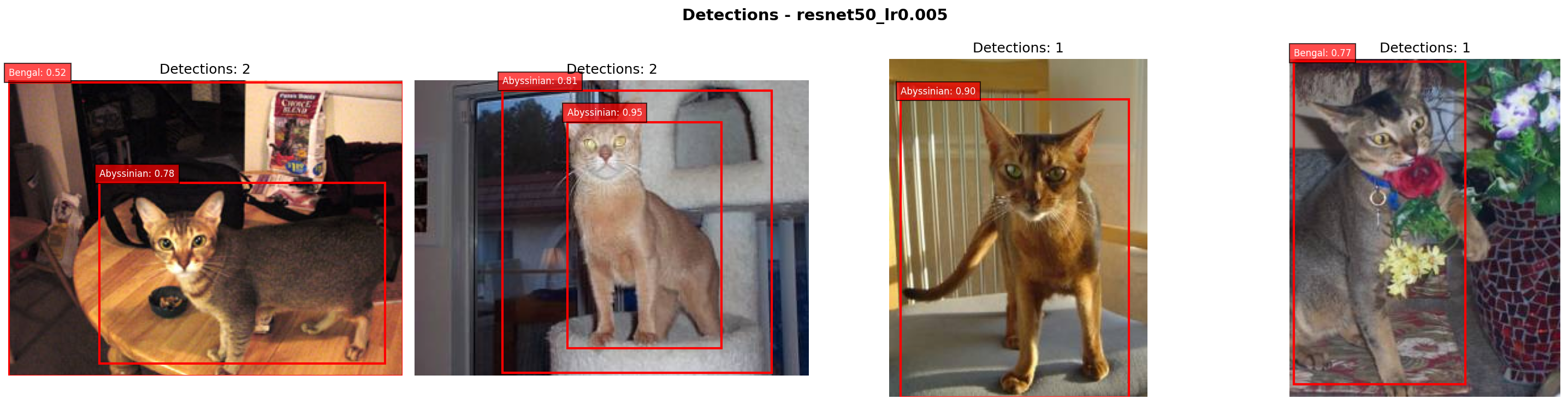
*Εικόνα 4.4: Learning rate vs detection accuracy*



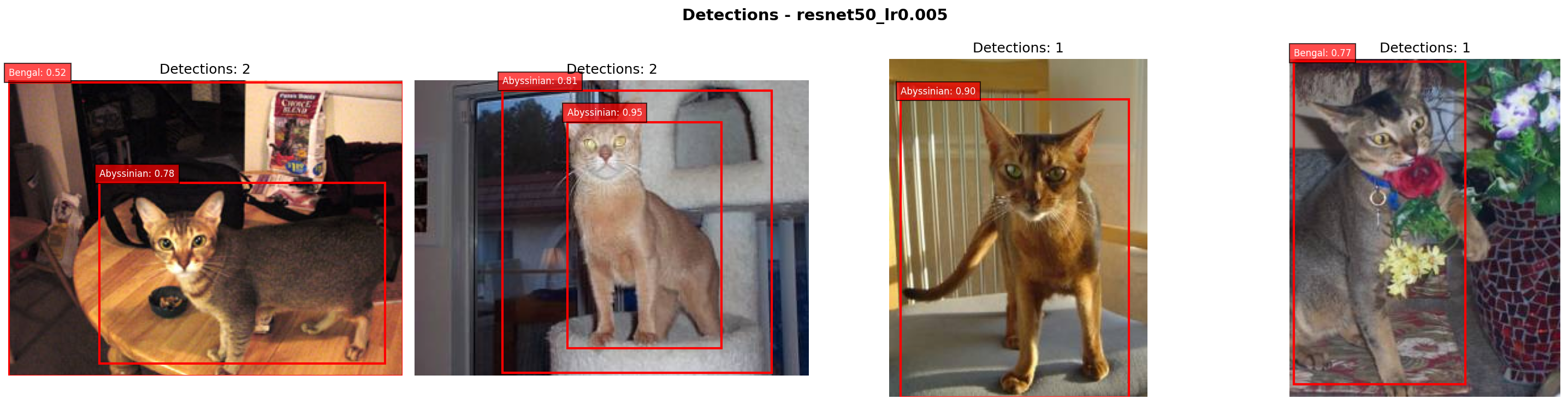
*Εικόνα 4.5: Χρόνος εκπαίδευσης vs detection accuracy*

Το ιδανικό lr εξαρτάται από το backbone: το ResNet50-FPN χρειάζεται υψηλότερο lr (0.005) για να προσαρμόσει τα πολλά στρώματα σε 4 epochs, ενώ το MobileNet λειτουργεί καλύτερα με χαμηλό (0.001).

## 4.5 Ενδεικτικά Αποτελέσματα Ανίχνευσης



*Εικόνα 4.6: Καλύτερο detection (ResNet50, lr=0.005)*



*Εικόνα 4.7: Δείγματα detections — ResNet50, lr=0.005*

## 4.6 Συμπεράσματα Άσκησης 4

* Το Faster R-CNN με ResNet50-FPN πετυχαίνει 83.18% detection accuracy σε μόλις 4 epochs — η δύναμη του pretrained COCO backbone.
* Το backbone καθορίζει το ιδανικό lr: ResNet50 θέλει 0.005, MobileNet θέλει 0.001. Η αντιστροφή τους ρίχνει σημαντικά την απόδοση.
* Το MobileNet είναι καλή επιλογή για γρήγορη εκπαίδευση (73.78% με lr=0.001), αλλά το ResNet50-FPN υπερτερεί κατά 10% με σωστό lr.
* Σε σύγκριση με την Άσκηση 2 (classification 90.87%), το detection είναι πιο δύσκολο (83.18%) γιατί απαιτεί και localization.

# Γενικά Συμπεράσματα

* Άσκηση 1: Το Vanilla CNN με SGD+cosine πετυχαίνει 56.43% στο CIFAR-100. Ο ρυθμός μάθησης είναι η πιο κρίσιμη υπερ-παράμετρος.
* Άσκηση 2: Το transfer learning φτάνει 90.87% με ResNet50 frozen. Τα pretrained features είναι εξαιρετικά ισχυρά.
* Άσκηση 3: Το U-Net πετυχαίνει mIoU=13.91% με κανονικό BG weight. Η μείωση του BG weight απαιτεί περισσότερες epochs.
* Άσκηση 4: Το Faster R-CNN με ResNet50-FPN πετυχαίνει 83.18% detection accuracy. Το backbone καθορίζει το ιδανικό lr.
* Γενικά: Η επιλογή optimizer και learning rate εξαρτάται από την αρχιτεκτονική και το task. SGD για απλά CNN, Adam για transfer learning, AdamW για segmentation, SGD για detection.