Παρουσίαση εργασιών στα Νευρωνικά δίκτυα

Ιωακείμ Ευαγγελινός, 4187

Βάση Δεδομένων: CIFAR10

Ιδιαιτερότητες βάσης δεδομένων:

Για την 1η και την 2η εργασία, οι εικόνες έχουν 20% πιθανότητα να έχουν ανεστραμμένα τα χρώματα, 50% πιθανότητα οριζόντιας περιστροφής και έχουν κανονικοποιηθεί. Στην 3η εργασία δεν έγιναν transformations για διευκόλυνσή μου όταν το δοκίμαζα εμφανίζοντας τις εικόνες.

Στόχοι:

Στην 1η , οι στόχοι των μοντέλων είναι η σωστή κατηγοριοποίηση των εικόνων όλων των κλάσεων.

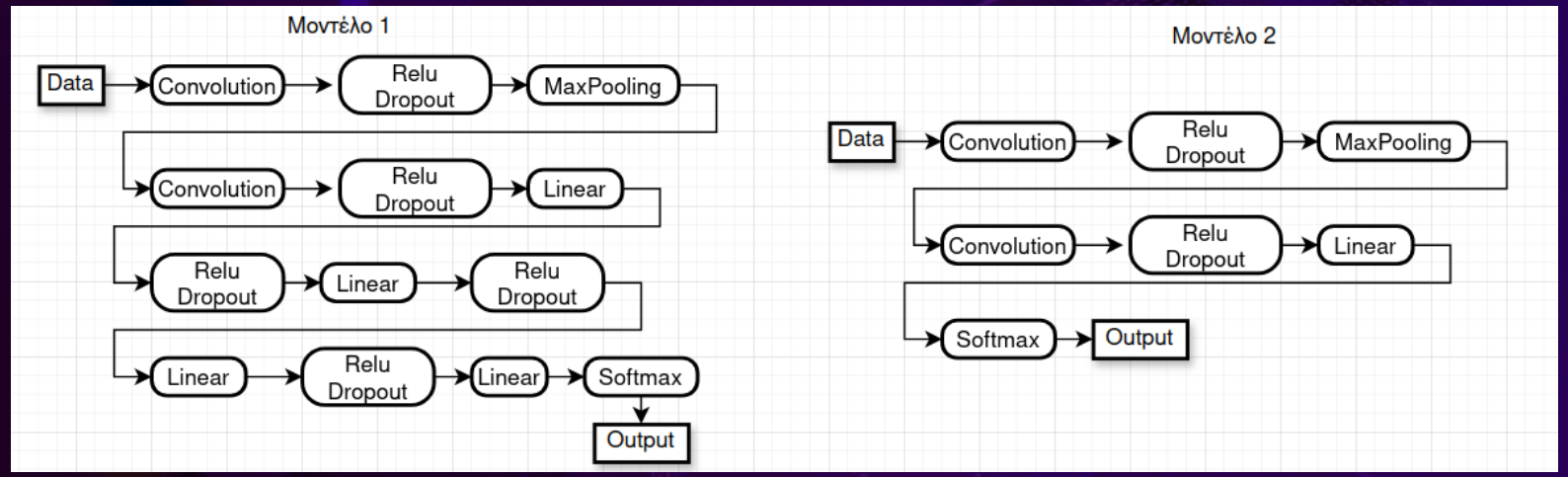
Στην 2η , οι στόχοι των μοντέλων είναι η σωστή κατηγοριοποίηση των εικόνων cat και ship.

Στην 3η εργασία ο στόχος είναι η αναδημιουργία εικόνων.

Όλα τα μοντέλα είναι φτιαγμένα είτε με την βιβλιοθήκη `torch.nn.Module` της Pytoch, είτε με την βιβλιοθήκη `sklearn`.

1η Εργασία (1/2)

Στην 1η εργασία, αρχικά συγκρίνω 2 μοντέλα με διαφορετικό αριθμό hidden layers και επέλεξα το καλύτερο (το 2ο):



Στη συνέχεια, συγκρίνω όλους του δυνατούς συνδυασμούς αυτής της αρχιτεκτονικής με τις συγκεκριμένες υπερ-παραμέτρους και τους συγκεκριμένους αριθμούς νευρώνων στα hidden layers.

Αριθμοί νευρώνων: [(100,70),(36,50)]
Learning rate: [0.001,0.0001]
Optimizers: [adam,sgd]

h1,h2	lr	optim	secs	test, train%
100,70	0.001	adam	912.9	61.05, 63.37
100,70	0.0001	adam	885.9	68.76, 74.19
36,50	0.001	adam	886.4	61.77, 63.72
36,50	0.0001	adam	882.6	65.46, 69.72
100,70	0.001	sgd	891.6	54.11, 55.26
100,70	0.0001	sgd	897.6	28.75, 28.35
36,50	0.001	sgd	888.3	48.25, 48.33
36,50	0.0001	sgd	890.9	24.38, 24.04

1η Εργασία (2/2)

Έπειτα συγκρίθηκαν οι κατηγοριοποιητές 1-Nearest Neighbor, 3-Nearest Neighbors και Nearest Centroid ως προς την ακρίβεια, τον χρόνο εκπαίδευσης και δοκιμής, και την μνήμη που δεσμεύει το μοντέλο

Clf	Train time	Test time	Acc	Mem
1-NN	0.303	104.171	0.3545	614800785
3-NN	0.275	104.166	0.3302	614800785
NC	0.501	0.26370	0.2775	246278

Από τα αποτελέσματα των κατηγοριοποιητών, συμπεράνω ότι ο πιο αποδοτικός κατηγοριοποιητής προς την ακρίβεια είναι ο 1-NN αλλά ο πιο αποδοτικό προς την χρήση πόρων είναι ο NC.

Το τελικό συμπέρασμα της 1ης εργασίας είναι ότι τα νευρωνικά μοντέλα έχουν πολύ καλύτερη απόδοση προς την ακρίβεια από τους κατηγοριοποιητές και θα χρησιμοποιούσα αυτά (συγκεκριμένα το 2ο) σε προβλήματα πραγματικού κόσμου.

2η Εργασία (1/2)

Στην 2η εργασία, συγκρίνω όλους του δυνατούς συνδυασμούς SVM με τις συγκεκριμένες υπερ-παραμέτρους (συνολικά 39 μοντέλα):
C = [1,0.1,0.001]
Kernel= [rbf, poly, linear, signmoid]
Gamma=["auto","scale",0.001]
Degree=[3,5]

Το καλύτερο μοντέλο είχε C=1, kernel=rbf, Gamma=auto και Degree=0 (αφού δεν παίζει ρόλο το degree στα rbf)

Table 1: SVM Comparison

Kernel	C	Tol	Gamma	Degree	Train_acc	Test_acc	Time_secs	Acc_cat	Acc_ship	Iterations	Converged
linear	1	0.001	0	0	0.5395	0.504	62.9274	0.504	0.504	8000	False
poly	1	0.001	auto	3	0.9577	0.683	119.6071	0.683	0.683	8000	False
poly	1	0.001	scale	3	0.8508	0.725	129.757	0.725	0.725	8000	False
poly	1	0.001	0.001	3	0.8842	0.592	92.948	0.592	0.592	8000	False
poly	1	0.001	auto	5	0.8483	0.555	108.538	0.555	0.555	8000	False
poly	1	0.001	scale	5	0.8759	0.707	145.6832	0.707	0.707	8000	False
poly	1	0.001	0.001	5	0.8128	0.5345	79.3355	0.5345	0.5345	8000	False
rbf	**1**	**0.001**	**auto**	**0**	**0.961**	**0.868**	**86.815**	**0.868**	**0.868**	**8000**	**False**
rbf	1	0.001	scale	0	0.9319	0.863	81.016	0.863	0.863	6749	True
rbf	1	0.001	0.001	0	0.9943	0.8245	161.5344	0.8245	0.8245	8000	False
sigmoid	1	0.001	auto	0	0.561	0.579	107.7995	0.579	0.579	5329	True
sigmoid	1	0.001	scale	0	0.5723	0.586	106.8842	0.586	0.586	6932	True
sigmoid	1	0.001	0.001	0	0.5134	0.5325	86.9114	0.5325	0.5325	3513	True
linear	0.1	0.001	0	0	0.5395	0.504	61.5824	0.504	0.504	8000	False
poly	0.1	0.001	auto	3	0.785	0.718	128.8261	0.718	0.718	8000	False
poly	0.1	0.001	scale	3	0.7183	0.6985	133.3068	0.6985	0.6985	6866	True
poly	0.1	0.001	0.001	3	0.9393	0.621	104.6276	0.621	0.621	8000	False
poly	0.1	0.001	auto	5	0.8892	0.7125	143.1497	0.7125	0.7125	8000	False
poly	0.1	0.001	scale	5	0.6727	0.6235	146.0265	0.6235	0.6235	8000	False
poly	0.1	0.001	0.001	5	0.8184	0.536	81.206	0.536	0.536	8000	False
rbf	0.1	0.001	auto	0	0.8252	0.8065	105.0626	0.8065	0.8065	4094	True
rbf	0.1	0.001	scale	0	0.8205	0.801	102.3941	0.801	0.801	3955	True
rbf	0.1	0.001	0.001	0	0.6846	0.67	141.5823	0.67	0.67	5175	True
sigmoid	0.1	0.001	auto	0	0.5621	0.58	117.8569	0.58	0.58	5631	True
sigmoid	0.1	0.001	scale	0	0.5801	0.5955	130.9691	0.5955	0.5955	7463	True
sigmoid	0.1	0.001	0.001	0	0.51	0.529	113.7114	0.529	0.529	4785	True
linear	0.001	0.001	0	0	0.7399	0.7055	100.1849	0.7055	0.7055	8000	False
poly	0.001	0.001	auto	3	0.6031	0.6005	143.9571	0.6005	0.6005	4821	True
poly	0.001	0.001	scale	3	0.56	0.561	148.15	0.561	0.561	4972	True
poly	0.001	0.001	0.001	3	0.7258	0.7015	133.1379	0.7015	0.7015	7179	True
poly	0.001	0.001	auto	5	0.5807	0.5765	145.9212	0.5765	0.5765	5854	True
poly	0.001	0.001	scale	5	0.5322	0.53	146.3314	0.53	0.53	5000	True
poly	0.001	0.001	0.001	5	0.896	0.5915	132.7398	0.5915	0.5915	8000	False
rbf	0.001	0.001	auto	0	0.6826	0.6845	156.1029	0.6845	0.6845	5000	True
rbf	0.001	0.001	scale	0	0.6986	0.7105	152.4645	0.7105	0.7105	5000	True
rbf	0.001	0.001	0.001	0	0.5339	0.533	152.679	0.533	0.533	5000	True
sigmoid	0.001	0.001	auto	0	0.6558	0.6735	156.4322	0.6735	0.6735	5060	True
sigmoid	0.001	0.001	scale	0	0.6528	0.6705	157.5546	0.6705	0.6705	5038	True
sigmoid	0.001	0.001	0.001	0	0.6347	0.6555	151.4138	0.6555	0.6555	5045	True

2η Εργασία (2/2)

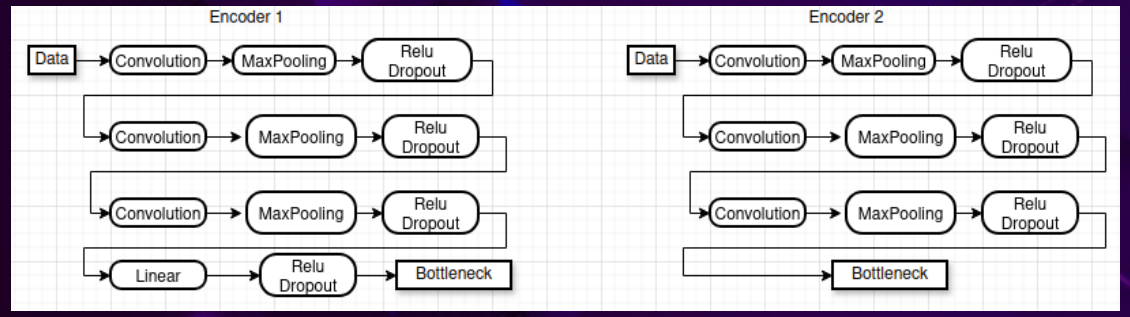
Στη συνέχεια, συγκρίνω το καλύτερο μοντέλο SVM με τους κατηγοριοποιητές 1-Nearest Neighbor, 3-Nearest Neighbors και Nearest Centroid, μαζί με ένα μοντέλο MLP με ένα hidden layer.

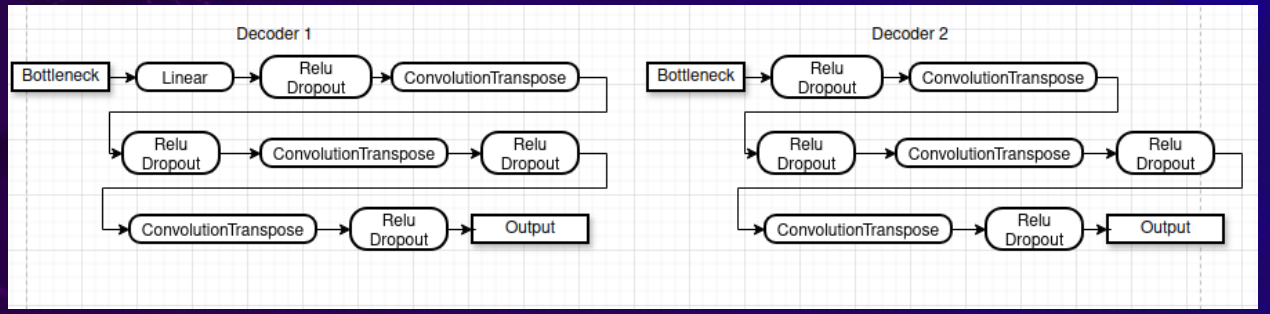
Classifier/Model	Train_time	Test_time	Accuracy
KNN(K=1)	0.1533	1.0619	0.747
KNN(K=3)	0.0562	0.7066	0.761
NC	0.10159	0.0299	0.6615
MLP(1 hidden)	80.0814	0.4209	0.7425
Best SVM	86.815	40.6294	0.868

Από τα αποτελέσματα των παραπάνω μοντέλων, συμπεράνω ότι το καλύτερο σε ακρίβεια μοντέλο είναι το SVM αλλά έχει υπερβολικά μεγαλύτερο χρόνο εξαγωγής συμπεράσματος (inference) σε σχέση με τα άλλα μοντέλα.

3η Εργασία (1/2)

Στην 3η εργασία, υλοποιήθηκε αναδημιουργία εικόνας με AutoEncoder. Αρχικά δημιούργησα 2 Encoders και 2 Decoders.





Δημιούργησα και ένα μοντέλο PCA με 512 components χρησιμοποιώντας την βιβλιοθήκη sklearn.

Το PCA που αναπτύχθηκε είχε loss 0.0026 και training time 14.38 δευτερόλεπτα

3η Εργασία (2/2)

Έπειτα έφτιαξα 4 AutoEncoders με τους συνδυασμούς των Encoder και Decode και συγκρίνω τα αποτελέσματά τους

Encoder(1/2)	Decoder(1/2)	Training Loss	Testing Loss	Training time	Testing time
1	1	16.023	0.128	3810.9	3.072
1	2	13.827	0.0109	3806.33	3.07
2	1	17.732	0.0147	1044.331	1.563
2	2	16.375	0.1396	971.516	1.604

Από τα παραπάνω συμπεράνω ότι το PCA με μεγάλο αριθμό components είναι η καλύτερη επιλογή για image reconstruction σε σχέση με τους autoencoders. Παρόλα αυτά, αν μειωθεί ο αριθμός των components στα 32, οι autoencoders είχαν καλύτερες επιδόσεις από το PCA.

Τελικά συμπεράσματα

Τα συμπεράσματα που εξάγω από όλη την έρευνα είναι τα εξής:

- Οι κατηγοριοποιητές έχουν χειρότερη απόδοση από τα νευρωνικά σε γενικές γραμμές στην κατηγοριοποίηση πολλών κλάσεων. Όταν η κατηγοριοποίηση γίνεται ανάμεσα σε 2 κλάσεις, τότε αποδίδουν σημαντικά καλύτερα και ανταγωνίζονται τα νευρωνικά δίκτυα με 1 hidden layer.
- Οι επιδόσεις των νευρωνικών δικτύων εξαρτιούνται σημαντικά από τον αριθμό των hidden layers, των νευρώνων και των υπερ-παραμέτρων.
- Η εκπαίδευση ενός νευρωνικού δικτύου είναι χρονοβόρα και υπολογιστικά απαιτητική.
- Τα SVM αποδίδουν καλύτερα από τα νευρωνικά δίκτυα με 1 hidden layer αλλά έχουν χρονοβόρο inference.
- Για την αναδημιουργία εικόνας, το PCA είναι πολύ καλό και χρησιμοποιήσιμο στον πραγματικό κόσμο, αν ο αριθμός των components είναι υψηλός. Αντιθέτως, αν είναι χαμηλός, τότε οι AutoEncoders έχουν καλύτερη απόδοση.