

Multiple Choice Questions

1. Ποιες από τις παρακάτω τεχνικές μπορούν να χρησιμοποιηθούν για τη μείωση της υπερπροσαρμογής (overfitting);
- Dropout και Χρήση Adam αντί Stochastic Gradient Descent.
 - Χρήση Adam αντί Stochastic Gradient Descent και Επαύξηση Δεδομένων.
 - Επαύξηση Δεδομένων και Dropout.
 - Κανένα από τα παραπάνω.

Υπερπροσαρμογή (Overfitting): Η υπερπροσαρμογή συμβαίνει όταν ένα νευρωνικό δίκτυο μαθαίνει πολύ καλά τα δεδομένα εκπαίδευσης, ακόμα και τον θόρυβο ή τις ιδιομορφίες τους, με αποτέλεσμα να έχει χαμηλή γενίκευση και κακή απόδοση σε νέα, άγνωστα δεδομένα.

Ο **Adam** και ο **SGD** είναι αλγόριθμοι βελτιστοποίησης που χρησιμοποιούνται για την εκπαίδευση νευρωνικών δικτύων. Ο Adam χρησιμοποιεί προσαρμοστικό learning rate και momentum, προσφέροντας γρήγορη και σταθερή σύγκλιση. Ωστόσο, καμία από αυτές τις μεθόδους δεν αποτελεί τεχνική regularization, επομένως δεν μειώνουν άμεσα την υπερπροσαρμογή.

Τεχνικές για Μείωση της Υπερπροσαρμογής:

- Dropout: Τυχαία απενεργοποίηση (drop) νευρώνων κατά την εκπαίδευση. Αποτρέπει τη συνεξάρτηση μεταξύ των νευρώνων → βοηθάει στη γενίκευση.
- Επαύξηση Δεδομένων (Data Augmentation): Χρήσιμο κυρίως σε εικόνες, ήχο, κείμενο. Δημιουργεί τεχνητά παραλλαγές των δεδομένων → λειτουργεί σαν αύξηση του dataset → μειώνει overfitting.
- Regularization (π.χ. L1, L2): Τιμωρεί μεγάλα weights στο μοντέλο.
- Early Stopping: Διακόπτει την εκπαίδευση όταν το validation loss σταματά να μειώνεται.
- Batch Normalization, Smaller Networks, Cross-Validation κ.λπ.

2. Ένα μοντέλο βαθιάς μάθησης που αναπτύξατε πετυχαίνει εξαιρετικά υψηλή ορθότητα (accuracy) (90%) στο σύνολο εκπαίδευσης (training set), αλλά σημαντικά χαμηλότερη (75%) στο σύνολο δοκιμής (test set). Για να αντιμετωπίσετε το πρόβλημα, αποφασίζετε να εφαρμόσετε ομαλοποίηση (regularization) τύπου L1 ή L2. Έχετε όμως βάσιμες υποψίες ότι κάποια από τα δεδομένα σας έχουν επισημειωθεί με εσφαλμένες ετικέτες (labels). Ποιον τύπο ομαλοποίησης θα προτιμήσετε να χρησιμοποιήσετε;

- L1
- L2
- Δεν έχει σημασία, τόσο η L1 όσο και η L2 είναι εξίσου κατάλληλες για τη συγκεκριμένη περίπτωση.
- Δεν μπορεί να αντιμετωπιστεί μέσω ομαλοποίησης το εν λόγω πρόβλημα.

υψηλή ακρίβεια στο training set } σημαδι overfit + δεδομένα με λάθος labels που εισάγουν θόρυβο στο training set
χαμηλή ακρίβεια στο test set

L1 regularization (aka Lasso)

- προσθέτει συνάρτηση κόστους σαν όρο: $\lambda \sum |w_i|$
- προκαλεί sparsity (μνένει/ει βάρη)
- ~ προτιμάται όταν $\exists$ πολλά άσχετα/θορυβώδη δεδομένα

L2 regularization (aka Ridge)

- προσθέτει συνάρτηση κόστους σαν όρο: $\lambda \sum w_i^2$
- μειώνει ομαλά όλα τα βάρη, συμπεριλαμβάνει όλα τα features
- ~ όλα τα δεδομένα έχουν σημασία

3. Κάνετε benchmarking στους χρόνους εκτέλεσης συχνά χρησιμοποιούμενων επιπέδων σε αρχιτεκτονικές συνελκτικών νευρωνικών δικτύων (CNN). Ποιο από τα παρακάτω επίπεδα αναμένετε να έχει τον μικρότερο χρόνο εκτέλεσης (σε floating point operations) για δεδομένη είσοδο;

- CONV layer (πράξη συνέλιξης και πρόσθεση όρου bias).
- MAX pooling layer.**
- Average pooling layer.
- Δεν έχουμε αρκετά δεδομένα για να απαντήσουμε.

Floating Point Operations (FLOPs) = πόσες πράξεις κινητής υποδιαστολής χρειάζονται (π.χ. προσθέσεις, πολλαπλασιασμοί). Δεν αφορά μόνο τον χρόνο σε hardware, αλλά και το θεωρητικό υπολογιστικό κόστος.

Επίπεδο (Layer)	Είδος πράξεων	Υπολογιστικό Κόστος (FLOPs)	Σχόλιο
CONV (Convolution)	Πολλαπλασιασμοί + αθροίσεις + bias	🔴 Πολύ Υψηλό	Βαρύτερο επίπεδο στα CNNs
Max Pooling	Συγκρίσεις	🟢 Πολύ Χαμηλό	Ελαφρύτερο όλων
Average Pooling	Αθροίσεις + διαίρεση	🟡 Χαμηλό	Ελαφρύ, αλλά βαρύτερο από Max Pooling
Fully Connected (FC)	Πολλαπλασιασμοί + αθροίσεις	🔴 Υψηλό	Ανάλογα με το μέγεθος των στρωμάτων

4. Τι από τα παρακάτω ισχύει για τους auto-encoders:

- Οι auto-encoders με ReLU activations και squared loss είναι ισοδύναμοι με PCA.
- Οι auto-encoders με linear activations και squared loss είναι ισοδύναμοι με PCA.**
- Οι auto-encoders με linear activations είναι ισοδύναμοι με PCA.
- Οι auto-encoders με ReLU activations είναι ισοδύναμοι με PCA.

Autoencoders είναι νευρωνικά δίκτυα που μαθαίνουν να κωδικοποιούν και να αποκωδικοποιούν τα δεδομένα, δηλαδή να τα "συμπίεζουν" και να τα ανακατασκευάζουν:

- Encoder: μειώνει τη διάσταση (παράγει latent representation).
- Decoder: προσπαθεί να ανακατασκευάσει την αρχική είσοδο.

Το **PCA** είναι μια γραμμική τεχνική μείωσης διαστάσεων που βρίσκει τις κατευθύνσεις μέγιστης διασποράς (principal components). Δεν είναι νευρωνικό δίκτυο, αλλά μαθηματική μέθοδος.

PCA $\equiv$ autoencoder με $\left\{ \begin{array}{l} \text{linear activation function} \\ \text{εκπαίδευση με square-loss function} \end{array} \right.$

5. Ο κωδικοποιητής (encoder) ενός transformer χρησιμοποιεί:

- a) Context-attention.
- b) Multi-head cross-attention.
- c) Multi-head self-attention.
- d) Τίποτα από τα παραπάνω.

Ο Encoder σε ένα Transformer αποτελείται από πολλαπλά επίπεδα που περιλαμβάνουν:

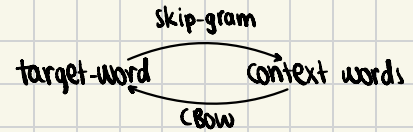
- 1. Multi-head self-attention
- 2. Feed-forward network
- 3. Residual connections και LayerNorm

Ο encoder ενός Transformer χρησιμοποιεί μηχανισμό multi-head self-attention, ώστε κάθε λέξη της εισόδου να λαμβάνει υπόψη της όλες τις άλλες λέξεις. Δεν χρησιμοποιείται cross-attention, καθώς αυτό αφορά τον decoder που επεξεργάζεται την έξοδο με βάση την είσοδο.

6. Ένα μοντέλο word2vec skip-gram εκπαιδεύεται με στόχο (target objective) να:

- a) Προβλέπει μία λέξη με βάση ένα παράθυρο n λέξεων γύρω από αυτή.
- b) Προβλέπει με βάση μία λέξη τα συμφοραζόμενά της (γειτονικές λέξεις) σε παράθυρο n λέξεων.
- c) Προβλέπει την απόσταση των διανυσμάτων μεταξύ γειτονικών λέξεων σε παράθυρο n .
- d) Τίποτα από τα παραπάνω.

Word2Vec



Το Word2Vec είναι μοντέλο που μαθαίνει ενσωματώσεις λέξεων (word embeddings). Υπάρχουν δύο βασικές παραλλαγές:

- CBOW: Προβλέπει τη λέξη-στόχο από τα συμφοραζόμενα
- Skip-gram: Προβλέπει τα συμφοραζόμενα (context words) από τη λέξη-στόχο (target word)

7. Εκπαιδεύετε ένα Generative Adversarial Network (GAN) για να παράγει ρεαλιστικές εικόνες αιλουροειδών. Ωστόσο, υποπτεύεστε ότι ο generator παρουσιάζει το πρόβλημα του mode collapse. Τι από τα παρακάτω μπορεί να αποτελεί ένδειξη για το συγκεκριμένο πρόβλημα;

- a) Η τιμή του κόστους (loss) του discriminator είναι υψηλή ενώ αυτή του generator παραμένει χαμηλή.
- b) Ο discriminator παρουσιάζει υψηλή ορθότητα (accuracy) στις αληθινές εικόνες, αλλά χαμηλή ορθότητα στις ψεύτικες.
- c) Ο generator παράγει διάφορες εικόνες άσχετες με αιλουροειδή.
- d) Ο generator παράγει μόνο εικόνες με λευκές τίγρεις

Ο Discriminator είναι το νευρωνικό δίκτυο μέσα σε ένα GAN που προσπαθεί να διακρίνει αν ένα δείγμα είναι αυθεντικό (προέρχεται από τα πραγματικά δεδομένα) ή τεχνητό (παραγόμενο από τον generator). Ο ρόλος του είναι να καθοδηγεί τον generator ώστε να δημιουργεί ρεαλιστικά δείγματα.

Το mode collapse σε ένα GAN εμφανίζεται όταν ο generator "κολλάει" και παράγει συνεχώς πολύ παρόμοιες ή ίδιες εικόνες, αγνοώντας τη φυσική ποικιλία του dataset.

8. Ποια από τις παρακάτω προτάσεις για τα μοντέλα διάχυσης (diffusion models) είναι εσφαλμένη;

- a) Η βασική αρχή λειτουργίας τους περιλαμβάνει μια προς-τα-εμπρός διαδικασία πρόσθεσης θορύβου και μια προς-τα-πίσω διαδικασία αφαίρεσης θορύβου.
- b) Η παραγωγή ρεαλιστικών δειγμάτων πραγματοποιείται κατά κύριο λόγο κατά τη προς-τα-εμπρός διαδικασία.
- c) Στην τυπική μορφή των diffusion models, η ανάστροφη (προς-τα-πίσω) διαδικασία αφαιρεί τον θόρυβο κατά κανόνα με χρήση αλυσίδων Markov.
- d) Τα diffusion models είναι γενικώς πιο αργά στην παραγωγή ρεαλιστικών δειγμάτων σε σχέση με τα GANs.

Τα diffusion models βασίζονται σε δύο στάδια:

- 1. Forward process ("διάχυση" / προσθήκη θορύβου): Παίρνουμε μια κανονική εικόνα και της προσθέτουμε σταδιακά τυχαίο θόρυβο, μέχρι να γίνει σχεδόν καθαρός θόρυβος (Gaussian noise).
- 2. Reverse process ("αποδιάχυση" / αφαίρεση θορύβου): Το μοντέλο μαθαίνει να αντιστρέφει τον θόρυβο βήμα-βήμα, ώστε από τυχαίο θόρυβο να παράγει ρεαλιστικά δείγματα (π.χ. εικόνες). Η αφαίρεση θορύβου βασίζεται σε διαδοχικά βήματα Markov.

GAN: 1 forward-pass ~ ανευθείας από latent vector

diffusion: 50-1000 reverse steps ~ σταδιακή απομάκρυνση από θόρυβο

9. Ποια από τις παρακάτω εργασίες δεν χρησιμοποιείται ως pretext task σε τεχνικές αυτό-επιβλεπόμενες μάθησης (self-supervised learning);

- a) Αναγνώριση γεωμετρικού μ/σ εικόνας.
- b) Ταξινόμηση καρέ από βίντεο στη σωστή σειρά.
- c) Σημασιολογική κατάτμηση εικόνας σε επίπεδο pixel.
- d) Κανένα από τα παραπάνω.

Η αυτό-επιβλεπόμενη μάθηση (self-supervised learning) είναι μια μέθοδος εκπαίδευσης μοντέλων χωρίς την ανάγκη χειροκίνητων ετικετών. Το μοντέλο μαθαίνει χρήσιμες αναπαραστάσεις μέσω ενός pretext task, δηλαδή μιας τεχνητής εργασίας που βασίζεται σε ιδιότητες των ίδιων των δεδομένων.

Σκοπός των pretext tasks είναι να ωθήσουν το δίκτυο να μάθει γενικές και χρήσιμες ιδιότητες (π.χ. σχήματα, αντικείμενα, κινήσεις), οι οποίες μπορούν στη συνέχεια να αξιοποιηθούν σε downstream εργασίες όπως ταξινόμηση ή ανίχνευση αντικειμένων.

Παραδείγματα pretext tasks:

- Αναγνώριση γεωμετρικού μετασχηματισμού εικόνας (π.χ. περιστροφή): μαθαίνει χωρικά χαρακτηριστικά.
- Ταξινόμηση καρέ βίντεο στη σωστή σειρά: μαθαίνει χρονική πληροφορία και δυναμική κίνησης.

Μη pretext task:

- Σημασιολογική κατάτμηση εικόνας σε επίπεδο pixel δεν είναι pretext task, καθώς απαιτεί πλήρως επισημειωμένα δεδομένα (pixel-wise labels). Πρόκειται για εποπτευόμενη εργασία που χρησιμοποιείται συνήθως ως downstream task, μετά την προεκπαίδευση του μοντέλου.

Open-ended Questions

Θέμα 1 (1.5 μονάδες)

1.1 Έστω συνελκτικό νευρωνικό δίκτυο (Convolutional Neural Network – CNN) που αποτελείται από τα επίπεδα που φαίνονται στην αριστερή στήλη του παρακάτω πίνακα. Συμπληρώστε τον πίνακα με τις διαστάσεις κάθε επιπέδου και το πλήθος των παραμέτρων του. Γράψτε τις διαστάσεις στη μορφή $W \times H \times C$, όπου W , H , C είναι το πλάτος (width), ύψος (height) και βάθος (channels). Όπου χρειάζεται, υποθέστε ότι padding και stride είναι ίσα με 1, εκτός αν προσδιορίζεται κάτι διαφορετικό.

Επεξήγηση συμβολισμών:

- CONV-N: συνελκτικό επίπεδο με N φίλτρα (πυρήνες) ύψους και πλάτους ίσων με x .
- POOL-n: max-pooling επίπεδο $n \times n$ με $\text{stride}=n$ και $\text{padding}=0$.
- FC-N: πλήρως συνδεδεμένο επίπεδο N νευρώνων.

#	level	dimensions	#parameters	
1	input	$32 \times 32 \times 3$	0	$\sim W_{in} \times H_{in} \times C_{in} = 32 \times 32 \times 3$
2	Conv-8	$32 \times 32 \times 8$	244	$\sim N=8$ kernels, $K \times K = 3 \times 3$, $S=P=1$ $W_{out} \times H_{out} \times D_{out} = W_{in} \times H_{in} \times N = 32 \times 32 \times 8$, #parameters = $N \cdot (K \cdot K \cdot C_{in} + 1) = 8 \cdot (3 \cdot 3 \cdot 3 + 1) = 8 \cdot 28 = 224$
3	ReLU	$32 \times 32 \times 8$	0	$\sim$ activation layer: δεν αλλάζει διαστάσεις, 0 parameters
4	POOL-2	$16 \times 16 \times 8$	0	$\sim$ παράθυρο $n \times n = 2 \times 2$ με $\text{stride } n=2$: $W_{out} \times H_{out} \times C_{out} = (W_{in}/n) \times (H_{in}/n) \times C_{in} = 16 \times 16 \times 8$
5	Conv-16	$16 \times 16 \times 16$	1168	$\sim N=16$ kernels, $K \times K = 3 \times 3$, $S=P=1$ $W_{out} \times H_{out} \times D_{out} = W_{in} \times H_{in} \times N = 16 \times 16 \times 16$, #parameters = $N \cdot (K \cdot K \cdot C_{in} + 1) = 16 \cdot (3 \cdot 3 \cdot 8 + 1) = 16 \cdot 73 = 1168$
6	ReLU	$16 \times 16 \times 16$	0	$\sim$ activation layer: δεν αλλάζει διαστάσεις, 0 parameters
7	Flatten	4096	0	$\sim N = W_{in} \times H_{in} \times C_{in} = 16 \cdot 16 \cdot 16 = 4096$, 0 parameters
8	FC-10	10	40970	$\sim$ input: $N=4096$ neurons, output: $M=10$ neurons, #parameters = $M \cdot (N+1) = 40970$
Σ			42362	

Convolutional layer, N kernels $K \times K$, stride S , padding= P :

$$W_{out} = \left\lfloor \frac{W_{in} - K + 2P}{S} \right\rfloor + 1, \quad H_{out} = \left\lfloor \frac{H_{in} - K + 2P}{S} \right\rfloor + 1, \quad C_{out} = N, \quad \#parameters = N \cdot (K \cdot K \cdot C_{in} + 1)$$

Activation layer: same dimensions, 0 parameters

Pool layer, $K \times K$ window, stride= S , padding= P :

$$W_{out} = \left\lfloor \frac{W_{in} - K + 2P}{S} \right\rfloor + 1, \quad H_{out} = \left\lfloor \frac{H_{in} - K + 2P}{S} \right\rfloor + 1, \quad C_{out} = C_{in}, \quad 0 \text{ parameters}$$

Flatten layer: $N_{out} = W_{in} \times H_{in} \times C_{in}$, 0 parameters

Fully-Connected layer FC-M: N =input neurons, M =output neurons, #parameters = $M \cdot (N+1)$

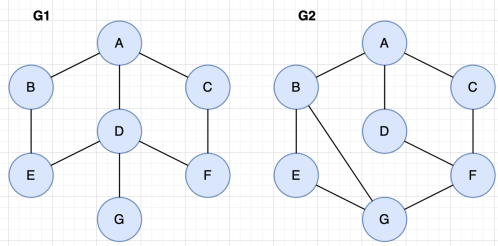
1.2 Περιγράψτε επιγραμματικά δύο λόγους για τους οποίους συνήθως είναι προτιμότερη η χρήση συνελκτικών επιπέδων αντί πλήρως συνδεδεμένων επιπέδων όταν δουλεύουμε με εικόνες.

- Εκμετάλλευση τοπικής πληροφορίας (locality): Τα φίλτρα σε ένα CONV layer εφαρμόζονται σε μικρές περιοχές της εικόνας (π.χ. 3×3), αξιοποιώντας τη χωρική δομή των δεδομένων — κάτι που τα FC αγνοούν πλήρως.
- Μείωση παραμέτρων $\rightarrow$ Αποδοτικότερο δίκτυο: Τα CONV layers έχουν επαναχρησιμοποιήσιμα φίλτρα, οπότε έχουν πολύ λιγότερες παραμέτρους σε σχέση με ένα FC layer που συνδέει κάθε pixel με κάθε επόμενο νευρώνα.
- Εκμετάλλευση συμμετρίας μετατοπίσεων (translation invariance): Επειδή το ίδιο φίλτρο εφαρμόζεται σε όλη την εικόνα, το δίκτυο μπορεί να αναγνωρίζει πρότυπα ανεξαρτήτως θέσης (π.χ. αν η γάτα είναι αριστερά ή δεξιά).
- Ευκολότερη γενίκευση – Λιγότερο overfitting: Λόγω του μικρότερου αριθμού παραμέτρων, το μοντέλο γενικεύει καλύτερα, χωρίς να υπερπροσαρμόζεται στα δεδομένα εκπαίδευσης.
- Κλιμακωτή αρχιτεκτονική (hierarchical feature learning): Με τη στοίβαξη CONV layers, το μοντέλο μαθαίνει χαμηλού επιπέδου χαρακτηριστικά (όπως άκρα) στις αρχικές στρώσεις, και υψηλού επιπέδου χαρακτηριστικά (π.χ. μάτια, αντικείμενα) στις βαθύτερες.

Χαρακτηριστικό	Συνελκτικά Επίπεδα (CONV)	Πλήρως Συνδεδεμένα (FC)
Τοπικότητα	Ναι – μαθαίνουν από μικρές περιοχές (π.χ. 3×3)	Όχι – συνδέουν κάθε pixel με κάθε νευρώνα
Αριθμός παραμέτρων	Μικρός – λόγω επαναχρησιμοποίησης φίλτρων	Πολύ μεγάλος – κάθε σύνδεση έχει βάρος
Μετατομιστική αμεταβλητότητα (translation invariance)	Παρουσιάζουν	Δεν παρουσιάζουν
Γενίκευση / Overfitting	Καλύτερη γενίκευση – λιγότερα βάρη	Υψηλός κίνδυνος overfitting
Ιεραρχική εξαγωγή χαρακτηριστικών	Μαθαίνουν επίπεδα αφηρημένων χαρακτηριστικών	Δεν έχουν ιεραρχία – επίπεδη πληροφορία
Καταλληλότητα για εικόνες	Ιδανικά	Όχι κατάλληλα

Θέμα 2 (1.0 μονάδα)

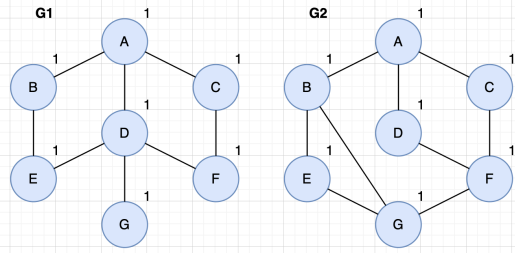
Δίνονται οι γράφοι G1 και G2 του παρακάτω σχήματος. Να υπολογιστεί η ομοιότητά τους με βάση τον πυρήνα Weisfeiler-Lehman (WL), δίνοντας όλα τα βήματα του αλγόριθμου.



Ο αλγόριθμος **Weisfeiler-Lehman (WL)** είναι μια μέθοδος που υπολογίζει ομοιότητα μεταξύ γράφων βασισμένη στη δομή τους. Ο πυρήνας WL εφαρμόζεται επαναληπτικά σε κάθε κόμβο του γράφου ως εξής:

1. Αρχικοποίηση: Όλοι οι κόμβοι παίρνουν ίδια ετικέτα (π.χ. "1").
2. Ενημέρωση: Σε κάθε επανάληψη: Κάθε κόμβος αποκτά νέα ετικέτα συνδυάζοντας την τρέχουσα ετικέτα του με τις ετικέτες των γειτόνων του (ταξινομημένες). Αυτή η συμβολοσειρά "χαρτογραφείται" (hashed) σε έναν ακέραιο αριθμό.
3. Feature vectors: Κάθε γράφος αποκτά ένα διάνυσμα καταμέτρησης των ετικετών ($\phi(G)$).
4. Kernel score: Υπολογίζουμε το εσωτερικό γινόμενο μεταξύ των διανυσμάτων των δύο γράφων.

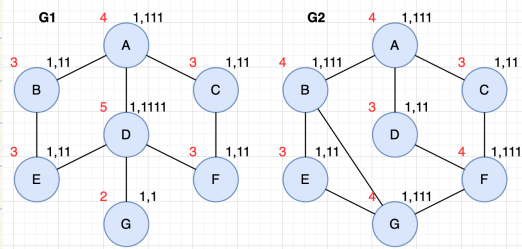
$i=0$:



label ID	$\phi(G1)$	$\phi(G2)$	product
1	7	7	49
Σ			49

$$\Rightarrow K^0(G1, G2) = 49$$

$i=1$:



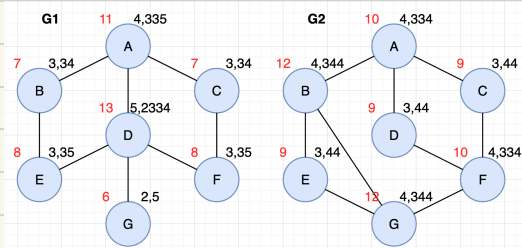
hash:

1,1-2
1,1-3
1,1-4
1,1-5

label ID	$\phi(G1)$	$\phi(G2)$	product
2	1	0	0
3	4	3	12
4	1	4	4
5	1	0	0
Σ			16

$$\Rightarrow K^1(G1, G2) = 16$$

$i=2$:



hash:

2,5-6
3,3,4-7
3,3,5-8
3,4,4-9
4,3,3,4-10
4,3,3,5-11
4,3,4,4-12
5,2,3,3,4-13

label ID	$\phi(G1)$	$\phi(G2)$	product
6	1	0	0
7	2	0	0
8	2	0	0
9	0	3	0
10	0	2	0
11	1	0	0
12	0	2	0
13	1	0	0
Σ			0

$$\Rightarrow K^2(G1, G2) = 0$$

Όλες οι ετικέτες ασύμβατες $\rightarrow$ σταματάμε

πυρήνας ομοιότητας:

$$K(G1, G2) = K^0(G1, G2) + K^1(G1, G2) + K^2(G1, G2) = 49 + 16 + 0 = 65$$

επιπλέον, ισχύει ότι: $\phi(G1) = [t(G1, c_1), t(G1, c_2), \dots, t(G1, c_{13})]$

c_i 1 2 3 4 5 6 7 8 9 10 11 12 13

$\phi(G1) = [7, 1, 4, 1, 1, 1, 2, 2, 0, 0, 1, 0, 1]$

$\phi(G2) = [7, 0, 3, 4, 0, 0, 0, 0, 3, 2, 0, 2, 0]$

$$K(G1, G2) = \phi(G1) \cdot \phi(G2) = 7 \cdot 7 + 4 \cdot 3 + 1 \cdot 4 = 49 + 12 + 4 = 65$$

Θέμα 3 (1.8 μονάδες)

Εξετάζεται το πρόβλημα που φαίνεται στο σχήμα, όπου δίνεται μια εικόνα και το αντικείμενο είναι να δημιουργηθεί μια περιγραφική λεζάντα φυσικής γλώσσας για την εικόνα. Την εικόνα επεξεργάζεται ένα CNN (δεν αναφέρεται και δε χρειάζεται για την επίλυση), με αποτέλεσμα 4 αναπαραστάσεις χαρακτηριστικών i_1, i_2, i_3, i_4 , όπου κάθε $i \in \mathbb{R}^2$, όπως φαίνεται στο σχήμα. Στη συνέχεια, η λεζάντα δημιουργείται αυτόματα με πλαθογέννηση από δίκτυο επαναληπτικών νευρώνων (RNN), που εξαρτάται από τις αναπαραστάσεις των χαρακτηριστικών της εικόνας.

3.1 Έστω η εικόνα εισόδου είναι αυτή που φαίνεται στο σχήμα, με τους ακόλουθους χάρτες χαρακτηριστικών εικόνας.

Σε κάθε χρονικό βήμα t , ο decoder που βασίζεται σε RNN λαμβάνει ως είσοδο ένα διάνυσμα $x_t \in \mathbb{R}^5$, το οποίο είναι η συνένωση (concatenation) της προηγούμενης εξόδου που ενσωματώνεται ως $y_{t-1} \in \mathbb{R}^3$ με μία αναπαράσταση εικόνας $c_t \in \mathbb{R}^2$ και χρησιμοποιεί αυτήν την είσοδο μαζί με την προηγούμενη κρυφή κατάσταση h_{t-1} για να υπολογίσει τη νέα κατάσταση h_t .

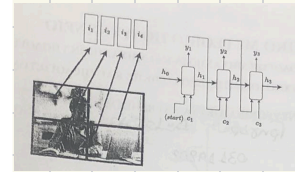
Αυτό ακολουθείται από ένα γραμμικό επίπεδο εξόδου (linear output layer) με πίνακα W_{yh} , όπου κάθε σειρά αυτού του πίνακα αντιστοιχεί στις λέξεις που αναφέρονται παραπάνω. Ο πίνακας input to hidden και ο πίνακας recurrence δίνονται αντίστοιχα από W_{hx} και W_{hh} . Ας υποθέσουμε ότι το RNN χρησιμοποιεί ενεργοποιήσεις ReLU, ότι όλα τα bias vectors είναι μηδενικά διανύσματα και ότι η αρχική κρυφή κατάσταση (hidden state) h_0 είναι ένα μηδενικό διάνυσμα.

Σε αυτήν την ερώτηση, υποθέστε ότι το $c_t := c$ είναι σταθερό για όλα τα χρονικά βήματα και προκύπτει από τη μέση συγκέντρωση των αναπαραστάσεων των χαρακτηριστικών εικόνας, χωρίς μηχανισμό προσοχής.

Υπολογίστε τις 3 πρώτες λέξεις της λεζάντας χρησιμοποιώντας greedy coding (επιλογή της πιο πιθανής λέξης σε κάθε βήμα).

embedding vectors:

$$\begin{aligned} y_{(\text{start})} &= [0, 0, 0]^T \\ y_{(\text{stop})} &= [1, 1, 1]^T \\ y_{\text{student}} &= [1, -2, 0]^T \\ y_{\text{reading}} &= [0, -1, -1]^T \\ y_{\text{teacher}} &= [-1, 2, 0]^T \\ y_{\text{studying}} &= [0, 2, 1]^T \end{aligned}$$



vocabulary $\sim$ βάζει μαζί με το indexing)

$$V = \{(\text{start}), (\text{stop}), \text{teacher}, \text{student}, \text{reading}, \text{studying}\}$$

feature maps:

$$i_1 = [4, 0]^T, \quad i_2 = [0, 4]^T, \quad i_3 = [0, 0]^T, \quad i_4 = [0, 0]^T$$

linear output layer:

$$W_{yh} = \begin{bmatrix} -5 & -5 & \# (\text{start}) \\ 0 & 3 & \# (\text{stop}) \\ 1 & 2 & \# \text{teacher} \\ 2 & 2 & \# \text{student} \\ 3 & -1 & \# \text{reading} \\ 2.9 & 0 & \# \text{studying} \end{bmatrix}$$

input-to-hidden:

$$W_{hx} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

hidden-to-hidden:

$$W_{hh} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

σταθερό context vector:

$$c := \bar{c} = \frac{1}{4} \sum_{j=1}^4 i_j = \text{μέσος όρος των } i_1, i_2, i_3, i_4$$

► context vector εικόνας: $c = \bar{c} = \frac{1}{4} \cdot \sum_{i=1}^4 i_i = \frac{1}{4} \cdot (i_1 + i_2 + i_3 + i_4) = \frac{1}{4} \cdot [4 \ 4]^T = [1 \ 1]^T$

► χρονική στιγμή $t=1$:

$$\text{-input vector: } \begin{cases} y_0 = y_{(\text{start})} = [0 \ 0 \ 0]^T \\ c = [1 \ 1]^T \end{cases} \Rightarrow x_1 = (y_0, c) = [0 \ 0 \ 0 \ 1 \ 1]^T$$

$$\text{-hidden vector: } \begin{cases} x_1 = [0 \ 0 \ 0 \ 1 \ 1]^T \\ h_0 = [0 \ 0]^T \end{cases} \Rightarrow h_1 = \text{ReLU}(W_{hx} \cdot x_1 + W_{hh} \cdot h_0) = \text{ReLU}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \text{ReLU}([1 \ 1]^T) \Rightarrow h_1 = [1 \ 1]^T$$

$$W_{hx} \cdot x_1 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} [0 \ 0 \ 0 \ 1 \ 1]^T = \begin{bmatrix} 10+00+10+11+01 \\ 00+10+10+01+11 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad W_{hh} \cdot h_0 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} [0 \ 0]^T = \begin{bmatrix} 0 \cdot 0 + 1 \cdot 0 \\ 1 \cdot 0 + 0 \cdot 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{-output vector: } o_1 = W_{yh} \cdot h_1 = \begin{bmatrix} -5 & 0 & 1 & 2 & 3 & 2.9 \\ -5 & 3 & 2 & 2 & -1 & 0 \end{bmatrix}^T [1] = [-10 \ 3 \ 3 \ 4 \ 2 \ 2.9]^T \sim \text{logits}$$

$$\text{-greedy decoding: } \max(o_1) = 4, \arg\max(o_1) = 3 \Rightarrow \text{αντιστοιχεί στο "student"} \sim V(3)$$

► χρονική στιγμή $t=2$:

$$\text{-input vector: } \begin{cases} y_1 = y_{(\text{student})} = [1 \ -2 \ 0]^T \\ c = [1 \ 1]^T \end{cases} \Rightarrow x_2 = (y_1, c) = [1 \ -2 \ 0 \ 1 \ 1]^T$$

$$\text{-hidden vector: } \begin{cases} x_2 = [1 \ -2 \ 0 \ 1 \ 1]^T \\ h_1 = [1 \ 1]^T \end{cases} \Rightarrow h_2 = \text{ReLU}(W_{hx} \cdot x_2 + W_{hh} \cdot h_1) = \text{ReLU}\left(\begin{bmatrix} 2 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = \text{ReLU}([3 \ 0]^T) \Rightarrow h_2 = [3 \ 0]^T$$

$$W_{hx} \cdot x_2 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} [1 \ -2 \ 0 \ 1 \ 1]^T = \begin{bmatrix} 11+0(-2)+10+11+01 \\ 01+1(-2)+10+01+11 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \quad W_{hh} \cdot h_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} [1 \ 1]^T = \begin{bmatrix} 0 \cdot 1 + 1 \cdot 1 \\ 1 \cdot 1 + 0 \cdot 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\text{-output vector: } o_2 = W_{yh} \cdot h_2 = \begin{bmatrix} -5 & 0 & 1 & 2 & 3 & 2.9 \\ -5 & 3 & 2 & 2 & -1 & 0 \end{bmatrix}^T [3] = [-15 \ 0 \ 3 \ 6 \ 9 \ 8.7]^T \sim \text{logits}$$

$$\text{-greedy decoding: } \max(o_2) = 9, \arg\max(o_2) = 4 \Rightarrow \text{αντιστοιχεί στο "reading"} \sim V(4)$$

► χρονική στιγμή $t=3$:

$$\text{-input vector: } \begin{cases} y_2 = y_{(\text{reading})} = [0 \ -1 \ -1]^T \\ c = [1 \ 1]^T \end{cases} \Rightarrow x_3 = (y_2, c) = [0 \ -1 \ -1 \ 1 \ 1]^T$$

$$\text{-hidden vector: } \begin{cases} x_3 = [0 \ -1 \ -1 \ 1 \ 1]^T \\ h_2 = [3 \ 0]^T \end{cases} \Rightarrow h_3 = \text{ReLU}(W_{hx} \cdot x_3 + W_{hh} \cdot h_2) = \text{ReLU}\left(\begin{bmatrix} 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix}\right) = \text{ReLU}([0 \ 2]^T) \Rightarrow h_3 = [0 \ 2]^T$$

$$W_{hx} \cdot x_3 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} [0 \ -1 \ -1 \ 1 \ 1]^T = \begin{bmatrix} 10+0(-1)+1(-1)+11+01 \\ 01+1(-1)+1(-1)+01+11 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad W_{hh} \cdot h_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} [3 \ 0]^T = \begin{bmatrix} 0 \cdot 3 + 1 \cdot 0 \\ 1 \cdot 3 + 0 \cdot 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$$\text{-output vector: } o_3 = W_{yh} \cdot h_3 = \begin{bmatrix} -5 & 0 & 1 & 2 & 3 & 2.9 \\ -5 & 3 & 2 & 2 & -1 & 0 \end{bmatrix}^T [0] = [-10 \ 6 \ 4 \ 4 \ -2 \ 0]^T \sim \text{logits}$$

$$\text{-greedy decoding: } \max(o_3) = 6, \arg\max(o_3) = 1 \Rightarrow \text{αντιστοιχεί στο "stop"} \sim V(1) \Rightarrow \text{τεματισμός πρότασης}$$

► Τελική λέξη: "student reading"

3.2 Ας υποθέσουμε τώρα ότι, αντί να χρησιμοποιεί ένα σταθερό c_t για όλα τα χρονικά βήματα, ο αποκωδικοποιητής που βασίζεται σε RNN έχει έναν μηχανισμό προσοχής scaled dot-product που παρακολουθεί τις αναπαραστάσεις των χαρακτηριστικών της εικόνας. Για κάθε χρονικό βήμα, το διάνυσμα ερωτήματος είναι h_{t-1} και οι αναπαραστάσεις των χαρακτηριστικών της εικόνας i_1, i_2, i_3, i_4 , χρησιμοποιούνται τόσο ως κλειδιά (key) όσο και ως τιμές (value). Υποθέστε ξανά ότι το h_0 είναι ένα μηδενικό διάνυσμα. Για το 1ο χρονικό βήμα ($t=1$), υπολογίστε τις πιθανότητες προσοχής (attention probabilities) και το διάνυσμα εικόνας που προκύπτει c_1 . Η 1η λέξη θα είναι ίδια ή διαφορετική από αυτήν στην προηγούμενη ερώτηση;

Scaled dot-product:

Query matrix: $Q = h_0 \in \mathbb{R}^{1 \times 2}$

Keys: $K = [i_1 \ i_2 \ i_3 \ i_4]^T \in \mathbb{R}^{4 \times 2}$

Values: $V = K \in \mathbb{R}^{4 \times 2}$

$d_k = 2$: διάσταση key-vectors

$\text{softmax}(x_i) = e^{x_i} / \sum e^{x_i}$

$$\textcircled{1} \text{ Attention Scores: } \text{Score} = Q \cdot K^T = \begin{bmatrix} 0 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\textcircled{2} \text{ Scaled Scores: } \text{Scaled Score} = \frac{\text{Score}}{\sqrt{d_k}} = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\textcircled{3} \text{ Attention Weights: } P = \text{softmax}(\text{Scaled Score}) = \text{softmax}(\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}) = \frac{1}{4} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

$$Z = \sum e^{x_i} = 4e^0 = 4, e^0 = 1$$

$$\textcircled{4} \text{ Context Vector: } c_1 = P \cdot V = \sum a_i \cdot i_i = 0.25 \cdot (i_1 + i_2 + i_3 + i_4) = 0.25 \cdot \begin{bmatrix} 4 & 4 \end{bmatrix}^T \rightarrow c_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$$

Συμπερασμα, $c_1 = c_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, άρα καταληγουμε συνν ίδια 1^η λέξη $\rightarrow$ 'student'

3.3 Υπολογίστε τη 2η λέξη της λεζάντας χρησιμοποιώντας τον αποκωδικοποιητή (decoder) RNN με χρήση προσοχής (attention).

Από (3.1) έχουμε ότι $Q = h_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$

$$\textcircled{1} \text{ Attention Scores: } \text{Score} = Q \cdot K^T = \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 4 & 0 & 0 \end{bmatrix}$$

$$\textcircled{2} \text{ Scaled Scores: } \text{Scaled Score} = \frac{\text{Score}}{\sqrt{d_k}} = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 4 & 4 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2.83 & 2.83 & 0 & 0 \end{bmatrix}$$

$$\textcircled{3} \text{ Attention Weights: } P = \text{softmax}(\text{Scaled Score}) = \text{softmax}(\begin{bmatrix} 2.83 & 2.83 & 0 & 0 \end{bmatrix}) = \frac{1}{3594} \cdot \begin{bmatrix} 16.97 & 16.97 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.472 & 0.472 & 0.0278 & 0.0278 \end{bmatrix}$$

$$Z = 2 \cdot 16.97 + 2 \cdot 1 = 35.94, e^{2.83} = 16.97, e^0 = 1$$

$$\textcircled{4} \text{ Context Vector: } c_2 = \sum a_i \cdot i_i = 0.472 \begin{bmatrix} 4 \end{bmatrix} + 0.472 \begin{bmatrix} 0 \end{bmatrix} + 0.0278 \begin{bmatrix} 0 \end{bmatrix} + 0.0278 \begin{bmatrix} 0 \end{bmatrix} = \begin{bmatrix} 1.888 & 1.888 \end{bmatrix}^T$$

χρονική στιγμή $t=2$:

$$\text{-input vector: } \begin{cases} y_1 = y_{\text{student}} = \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}^T \\ c_2 = \begin{bmatrix} 1.888 & 1.888 \end{bmatrix}^T \end{cases} \Rightarrow x_2 = (y_1, c_2) = \begin{bmatrix} 1 & 2 & 0 & 1.888 & 1.888 \end{bmatrix}^T$$

$$\text{-hidden vector: } \begin{cases} x_2 = \begin{bmatrix} 1 & 2 & 0 & 1.888 & 1.888 \end{bmatrix}^T \\ h_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T \end{cases} \Rightarrow h_2 = \text{ReLU}(W_{nx} \cdot x_2 + W_{nh} \cdot h_1) = \text{ReLU}\left(\begin{bmatrix} 2.9 \\ -0.1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = \text{ReLU}(\begin{bmatrix} 3.9 & 0.9 \end{bmatrix}) \Rightarrow h_2 = \begin{bmatrix} 3.9 & 0.9 \end{bmatrix}^T$$

$$W_{nx} \cdot x_2 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 & 1.888 & 1.888 \end{bmatrix}^T = \begin{bmatrix} 1 \cdot 1 + 0 \cdot 2 + 1 \cdot 0 + 1 \cdot 1.888 + 0 \cdot 1.888 \\ 0 \cdot 1 + 1 \cdot 2 + 1 \cdot 0 + 0 \cdot 1.888 + 1 \cdot 1.888 \end{bmatrix} = \begin{bmatrix} 2.9 \\ -0.1 \end{bmatrix}$$

$$\text{-output vector: } o_2 = W_{oh} \cdot h_2 = \begin{bmatrix} -5 & 0 & 1 & 2 & 3 & 2.9 \\ -5 & 3 & 2 & 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} 3.9 \\ 0.9 \end{bmatrix} = \begin{bmatrix} -23.88 & 26.64 & 5.664 & 9.552 & 10.776 & 11.275 \end{bmatrix}$$

$$\text{-greedy decoding: } \max(o_2) = 11.275, \text{argmax}(o_2) = 5 \Rightarrow \text{αντιστοιχεί στο 'studying' } \sim V(5)$$

Συμπέρασμα:

Ο μηχανισμός προσοχής επιτρέπει στον αποκωδικοποιητή να επικεντρώνεται επιλεκτικά σε διαφορετικές περιοχές της εικόνας ανά χρονικό βήμα, οδηγώντας σε πιο εκφραστική και ακριβή λεζάντα, αναδεικνύοντας τη δυναμική προσαρμοστικότητα του αντίθετη με τα στατικά context vectors.