Άσκηση 3
Εφαρμογές της Ψηφιακής Επεξεργασίας Σημάτων

Περιεχόμενα

Ο κώδικας μπορεί να βρεθεί εδώ:

https://github.com/GrigorisTzortzakis/Applications-of-Digital-Signal-Processing/tree/main/Exercise%203

ADSP-KSVD

Διαδικασία

1)

Αρχικά, για εικόνες έχουμε χρησιμοποιήσει αυτές που μας δίνονται. Επιπλέον, πρακτικά από κώδικα χρησιμοποιούμε τα έτοιμα function που δίνονται και το μόνο που κάνουμε εμείς είναι να φτιάξουμε ένα νέο main script, ώστε να εκτελεί όλα τα ζητούμενα.

Στο επόμενο βήμα καθορίζονται οι παράμετροι του αλγορίθμου:

- patchSize=8
- numAtoms=100
- T0=10
- err=1e-4
- numEpochs=100
- scaleF=0.25
- stride=16

Όλοι οι παράμετροι έχουν επιλεχθεί με τρόπο τέτοιο ώστε να έχουμε αποδεκτά αποτελέσματα στην ανακατασκευή ενώ ταυτόχρονα η όλη διαδικασία γίνεται σε σύντομο χρονικό διάστημα, δηλαδή σε κάτω από 15 λεπτά.

Ξεκινώντας με το patchSize, κάθε patch είναι ένα τετράγωνο 8×8 pixel, δηλαδή ένα διάνυσμα 64 διαστάσεων. Το 8×8 είναι τυπικό μέγεθος σε πολλούς αλγορίθμους (π.χ. JPEG), γιατί απομονώνει τοπικά χαρακτηριστικά (άκρα, γωνίες) χωρίς να αυξάνει υπερβολικά τον αριθμό διαστάσεων. Με μικρότερο patch (π.χ. 4×4) χάνουμε ικανότητα αναπαράστασης πιο απλών χαρακτηριστικών, ενώ με μεγαλύτερο (π.χ. 16×16) αυξάνεται εκθετικά ο υπολογιστικός φόρτος (16×16=256 διαστάσεις).

Προχωρώντας με το numAtoms, το μέγεθος του λεξικού K=100 επιλέγεται ώστε να είναι υπερπλήρες σε σχέση με τη διάσταση 64 (8×8). Ένα τέτοιο λεξικό (K>d) επιτρέπει πιο ευέλικτες, αραιές αναπαραστάσεις, καθώς υπάρχουν περισσότερα “άτομα” (basis vectors) για να διαλέξει ο OMP. Η τιμή 100 είναι ένα εύρος κατανεμημένο αρκετά πάνω από τις 64 διαστάσεις και ταυτόχρονα δεν είναι πολύ μεγάλο ώστε να μην χρειάζεται τεράστιους υπολογιστικούς πόρους και χρόνο για τον υπολογισμό.

Συνεχίζοντας, βάζουμε T0=10 ώστε να έχουμε μεγάλο sparsity, αφού αυτό ορίζει πόσους μη μηδενικούς συντελεστές έχουμε για να αναπαραστήσουμε κάθε patch. Συγκεκριμένα, έχουμε αναλογία περίπου 10% μη μηδενικών συντελεστών σε σχέση με τα 100 άτομα. Προσθέτοντας, ορίζουμε μέγιστο επιτρεπτό σφάλμα για τον Omp 1e-4.

Παρακάτω, επιλέγουμε 100 εποχές ώστε να υπάρχει ισορροπία ανάμεσα στον χρόνο εκτέλεσης και στην απόδοση του συστήματος. Παρατηρήθηκε μέσα από τα πειράματα ότι συγκλίνει το σύστημα σε αυτό τον αριθμό epoch, άρα δεν κερδίζουμε κάτι ουσιαστικό βάζοντας παραπάνω.

Τέλος, έχουμε scale=0.25, άρα έχουμε down-sampling των εικόνων στο 25% των αρχικών διαστάσεων ώστε να μειώσουμε δραστικά τον αριθμό των pixels και άρα των patches, χωρίς να χάνουμε τα κυριότερα γεωμετρικά χαρακτηριστικά. Με stride=16 σημαίνει ότι έχουμε 16 pixel απόσταση ανάμεσα στα patches και άρα έχουμε πιο απομακρυσμένα δείγματα.

Ας δούμε τώρα το κύριο μέρος του συστήματος. Φτιάχνουμε την λίστα train_list που περιέχει όλα τα αρχεία .jpg του φακέλου εκπαίδευσης. Για να υπολογιστεί εκ των προτέρων το μέγιστο πλήθος patches ανά εικόνα, διαβάζεται μόνο η πρώτη εικόνα της λίστας, μετατρέπεται σε gray scale και γίνεται κανονικοποίηση σε τύπο διπλής ακρίβειας και down-sampling στο προκαθορισμένο ποσοστό. Από τις διαστάσεις της προσαρμοσμένης εικόνας (Ύψος×Πλάτος) υπολογίζεται το πόσα πλήρη παράθυρα 8×8 χωράνε οριζόντια και κάθετα με βήμα ίσο με το stride και το γινόμενο αυτών των δύο αριθμών πολλαπλασιασμένο με τον συνολικό αριθμό εικόνων δίνει το μέγιστο πλήθος patches maxPer. Αμέσως μετά δημιουργείται ένας πίνακας Y διαστάσεων [patchSize² × maxPer×NumImages] γεμάτος μηδενικά.

Στη συνέχεια, το επιλεγμένο τετράγωνο τμήμα αναδιπλώνεται σε διάνυσμα 64×1 και υπολογίζεται η Ευκλείδεια νόρμα του. Εάν αυτή υπερβαίνει ένα όριο (1e–8), τότε το διάνυσμα διαιρείται δια της νόρμας του, ώστε να έχει μήκος μονάδα και αντιγράφεται στην τρέχουσα στήλη col του Y. Ο μετρητής col αυξάνεται κατά ένα. Αν η νόρμα είναι σχεδόν μηδέν, το patch θεωρείται ανεπαρκές και παραλείπεται, ώστε να μην γεμίζουν οι στήλες με άχρηστα δεδομένα. Στο τέλος όλων των εικόνων και θέσεων, οι περιττές πλέον στήλες κόβονται με μία μόνο εντολή, Y = Y(:,1:col-1), παράγοντας έναν συμπαγή πίνακα μόνο με ουσιαστικά patches.

Αμέσως μετά την κατασκευή του Y, ορίζεται τυχαία το αρχικό λεξικό D και γίνεται μηδέν ο πίνακας συντελεστών X. Η συνάρτηση DictionaryLearning λαμβάνει ως είσοδο το λεξικό D, τον πίνακα δεδομένων Y, την ανοχή

σφάλματος, το επιθυμητό sparsity, τον αριθμό εποχών και τον αρχικό X. Τότε, επιστρέφει το διάνυσμα των μέσων τετραγωνικών σφαλμάτων ανά εποχή, το ενημερωμένο λεξικό και τους συντελεστές.

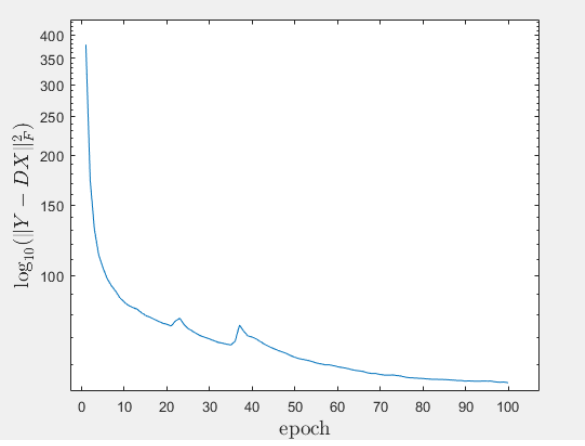
Στην τελευταία φάση testing, οι εικόνες από τον φάκελο ελέγχου διαβάζονται ξανά, επεξεργάζονται με τον ίδιο τρόπο (grayscale, down-sampling) και χωρίζονται σε διαδοχικά, μη επικαλυπτόμενα patches. Για κάθε patch υπολογίζεται η ευκλείδεια νόρμα και εάν αυτή είναι σχεδόν μηδέν, ορίζεται σε 1 για να αποφευχθεί διαίρεση με το μηδέν. Το patch κωδικοποιείται αραιά με την GenOMP, χρησιμοποιώντας το ήδη εκπαιδευμένο λεξικό, παράγοντας τους συντελεστές coeff. Η ανακατασκευασμένη τιμή του patch προκύπτει πολλαπλασιάζοντας το λεξικό επί τους συντελεστές και πολλαπλασιάζοντας εκ νέου με τη νόρμα, ώστε να επανέλθει η αρχική κλίμακα φωτεινότητας. Κάθε ανακατασκευασμένο παράθυρο τοποθετείται στη θέση του μέσα σε έναν πίνακα Irec.

2)

Αρχικά, επιδιώκουμε να ελαχιστοποιήσουμε την ολική συνάρτηση κόστους:

$$\min_{D,X} ||Y - DX||_F^2$$

διασφαλίζοντας ότι κάθε βήμα sparse-coding (GenOMP) και dictionary-update (K-SVD) μειώνει ή τουλάχιστον δεν αυξάνει σημαντικά το υπόλοιπο σφάλμα. Η θεωρία προβλέπει ότι με κάθε εποχή το λεξικό προσαρμόζεται καλύτερα στα δεδομένα και η τιμή της νόρμας Frobenius μειώνεται σταθερά.



Στο συγκεκριμένο διάγραμμα παρατηρούμε ότι στην πρώτη δεκάδα εποχών η πτώση του σφάλματος είναι ιδιαίτερα απότομη, από πολύ υψηλή τιμή (περίπου 400) σε τιμές της τάξης των 100–150. Μεταξύ των εποχών 10 και 30 η καμπύλη συνεχίζει να κατεβαίνει αλλά με μικρότερο ρυθμό και το περίεργο

είναι ότι εμφανίζονται δύο διακριτά “κύματα” ελαφράς αύξησης του σφάλματος, γύρω στις εποχές 25 και 35. Αυτό συμβαίνει επειδή στον K-SVD, δεν έχουμε εγγύηση μείωσης του συνολικού σφάλματος σε κάθε εποχή. Το λεξικό D και οι συντελεστές X ενημερώνονται διαδοχικά, δηλαδή πρώτα «σταθεροποιούμε» το D και βρίσκουμε το X με OMP, οπότε το σφάλμα κατεβαίνει. Έπειτα, κατά το βήμα K-SVD, ανανεώνουμε σειριακά κάθε άτομο d_k μέσω rank-1 SVD. Συμπερασματικά, τα ανοδικά «κύματα» οφείλονται στο ότι τα τοπικά rank-1 SVD updates των atoms κατά το dictionary-update μπορούν προσωρινά να αυξήσουν το συνολικό σφάλμα πριν την επαναπροσαρμογή των συντελεστών.

Προσθέτοντας, μετά την εποχή 40 η συνολική πτωτική τάση επανέρχεται, με πιο ομαλές αλλά ολοένα πιο μικρές βελτιώσεις και προς το τέλος των 100 εποχών, η καμπύλη πλησιάζει ένα οριζόντιο επίπεδο, υποδηλώνοντας ότι έχουμε φτάσει σε προσωρινή σύγκλιση. Οφείλουμε να τονίσουμε ότι η σύγκλιση δεν συμβαίνει στα 0 db αλλά στα 53 db, καθώς το μέτρο σφάλματος που βλέπουμε δεν είναι ο μέσος όρος ανά pixel ή patch, αλλά το συνολικό άθροισμα των τετραγώνων σε όλα τα patches, δηλαδή:

$$\boldsymbol{MSE} = \sum_{\boldsymbol{n=1}}^{\boldsymbol{N}} \left|\left|\boldsymbol{y_n} - \boldsymbol{D_{x_n}}\right|\right|_{\boldsymbol{2}}^{\boldsymbol{2}}$$

όπου κάθε yn έχει νόρμα 1 (είναι κανονικοποιημένο patch). Στην πρώτη εποχή, λόγω του τυχαίου λεξικού και των μηδενικών συντελεστών, η ανακατασκευή DX δίνει σχεδόν μηδέν, οπότε το σφάλμα ισούται με N. Για N≈400patches, ξεκινάμε γύρω στο 400.

Καθώς προχωρά η εκπαίδευση, ο OMP βρίσκει έως 10 atom για κάθε patch αλλά συχνά σταματάει λόγω sparsity (και όχι λόγω επίτευξης απόλυτης ανοχής), αφήνοντας ένα υπόλοιπο που δεν μπορεί να μηδενιστεί. Στο τέλος, όταν όλες οι εποχές ολοκληρωθούν, το λεξικό έχει «μάθει» να αναπαριστά τα patches όσο πιο πιστά γίνεται μέσα στον περιορισμό των 10 συντελεστών. Όμως, τα λεπτότερα μοτίβα και ο θόρυβος παραμένουν ως υπόλοιπα και αθροίζοντας όλα αυτά τα τετραγωνικά residuals έχουμε περίπου 53 MSE. Αν υπήρχε η δυνατότητα για καλύτερο μηχάνημα για την υλοποίηση του κώδικα τότε σίγουρα θα είχαμε πολύ μικρότερο mse στο τέλος, αλλά την τωρινή στιγμή αυτό απαιτεί πάνω από 10 ώρες για τον υπολογισμό (8+ ώρες με MacBook m3, 15+ ώρες με i5 6600).

Συμπερασματικά, πετυχαίνουμε ικανοποιητικό μέσο σφάλμα/pixel. Είναι 53, που είναι το συνολικό σφάλμα προς τον συνολικό αριθμό patch N (400). Τότε, έχουμε περίπου 0.13 τετραγωνικό σφάλμα ανά patch.

3)

Στο αριστερό μέρος, όπου απεικονίζεται το αρχικό λεξικό, κάθε atom είναι αποτέλεσμα τυχαίας κανονικοποιημένης κατανομής Gaussian. Ως εκ τούτου, η συνολική εμφάνιση θυμίζει λευκό θόρυβο χωρίς καμία διακριτή δομή ή προσανατολισμό. Δεν παρατηρούνται ευθείες, γωνίες ή κλασικά στοιχεία. Κάθε υπο-εικόνα μοιάζει με ανεξάρτητο τυχαίο μοτίβο. Αυτή η τυχαιότητα εξασφαλίζει ότι το λεξικό ξεκινά χωρίς bias προς συγκεκριμένα χαρακτηριστικά των εικόνων εκπαίδευσης, αλλά ταυτόχρονα δεν προσφέρει κανένα χρήσιμο θεμελιώδες σχήμα.

Στο δεξιό μέρος, όπου απεικονίζεται το λεξικό μετά την εκπαίδευση, διακρίνουμε σαφώς δομημένα μοτίβα. Συγκεκριμένα, βλέπουμε ευθύγραμμες ακμές σε διάφορες διευθύνσεις, γωνίες και απλά γεωμετρικά στοιχεία. Κάποια άτομα θυμίζουν οριζόντιες ή κάθετες γραμμές (edge-detectors), ενώ λίγα μοτίβα απεικονίζουν πιο σύνθετους συνδυασμούς ακμών. Αυτή η δομή μαρτυρά ότι ο αλγόριθμος KSVD, σε συνδυασμό με την αραιή κωδικοποίηση GenOMP, έχει εξάγει τα πιο αντιπροσωπευτικά χαρακτηριστικά των training patches.

Συμπερασματικά, η μετάβαση από το θορυβώδες αρχικό λεξικό στο δομημένο τελικό λεξικό επιβεβαιώνει την επιτυχία της διαδικασίας μάθησης, αφού το λεξικό δεν καλύπτει πια τυχαία το χώρο των πιθανών διανυσμάτων, αλλά επικεντρώνεται σε εκείνες τις διευθύνσεις που παρουσιάζουν συχνότητα και σημασία στα δεδομένα εικόνων.

4)

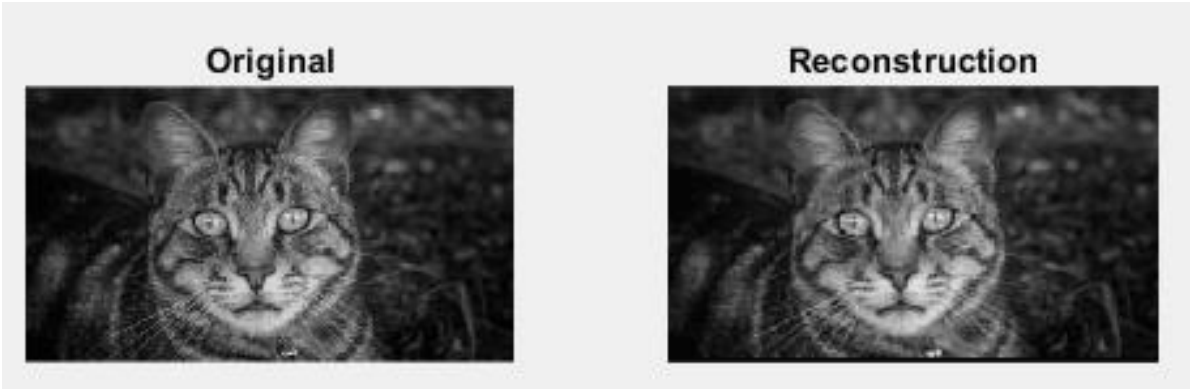
```
Reconstruction MSE στο testing set:
 cat_known.jpg   0.02342
 cat_unknown.jpg 0.06678
 field.jpg       0.18737
 floathouse.jpg  0.11654
 keanu.jpg       0.02396
 parrot.jpg      0.10190
 woman.jpg       0.18141
```

Στη διαδικασία δοκιμής παρατηρείται ότι οι εικόνες με σχετικά απλή δομή και συχνά επαναλαμβανόμενα μοτίβα ανακατασκευάζονται με πολύ μικρό σφάλμα. Συγκεκριμένα, τόσο η «cat_known» (MSE ≈ 0.0234) όσο και η «keanu» (MSE ≈ 0.0240) παρουσιάζουν τα χαμηλότερα σφάλματα, γεγονός που υποδεικνύει πως το λεξικό είναι ιδιαίτερα ικανό να αναπαραστήσει μοτίβα που είτε υπήρχαν στο σετ εκπαίδευσης είτε είναι παρόμοιας υφής, όπως οι ομοιόμορφες επιφάνειες τρίχας και δέρματος. Αντίθετα, εικόνες με πλουσιότερη υφή, όπως το «field» (MSE ≈ 0.1874) και το «woman» (MSE ≈ 0.1814), δείχνουν σημαντικά υψηλότερα σφάλματα, αφού η ποικιλία λεπτομερειών και οι αραιές δομές του γρασιδιού ή των λεπτών χαρακτηριστικών του προσώπου δυσχεραίνουν την πιστή αναπαραγωγή τους με περιορισμένο αριθμό ατόμων στο λεξικό.

Επίσης, είναι ενδιαφέρον η σύγκριση της «cat_unknown.jpg» με την «cat_known.jpg». Παρόλο που η άγνωστη γάτα έχει κάπως παρόμοια δομή με αυτή που ενσωματώθηκε στο σύνολο εκπαίδευσης, εμφανίζει περίπου τριπλάσιο σφάλμα σε σχέση με τη γνωστή. Το αποτέλεσμα αυτό αναδεικνύει την ευαισθησία της μεθόδου σε μικρές διαφοροποιήσεις στα μοτίβα. Οι νέες λεπτομέρειες και διαφορετικές τιμές φωτεινότητας αυξάνουν το σφάλμα όσο δεν περιλαμβάνονται ρητά στο λεξικό. Αντίστοιχα, το «parrot» (MSE ≈ 0.1019) και το «floathouse» (MSE ≈ 0.1165) κατατάσσονται σε μεσαία ζώνη σφάλματος, καθώς τα έντονα χρώματα των φτερών και οι ευθείες γραμμές απαιτούν επιπλέον συνιστώσες για ακριβέστερη αναπαραγωγή.

Η πιο ενδιαφέρον παρατήρηση είναι το χαμηλό σφάλμα που έχει η εικόνα «keanu». Παρά το γεγονός ότι το σύστημα δεν έχει δει ποτέ ανθρώπινα πρόσωπα κατά την εκπαίδευση, η εικόνα δίνει εκπληκτικά μικρό σφάλμα ανακατασκευής γιατί τα στατιστικά σε επίπεδο patch ευθυγραμμίζονται πολύ καλά με τα atom που έμαθε το λεξικό από τις εικόνες των γατών. Αντιθέτως, στην εικόνα «cat_unknown», παρόλο που απεικονίζει επίσης μια γάτα, υπάρχουν πράγματα που απλώς δεν υπήρχαν στο σετ εκπαίδευσης.

Η γάτα που υπήρχε στο σετ εκπαίδευσης ανακατασκευάζεται σχεδόν τέλεια, αφού οι ομαλές μεταβάσεις στο τρίχωμα, οι αντιθέσεις και η λεπτομέρεια στα μάτια διατηρούνται με ελάχιστο θόρυβο. Το αποτέλεσμα δείχνει καθαρή αναπαραγωγή των μοτίβων που το λεξικό έχει ήδη μάθει.

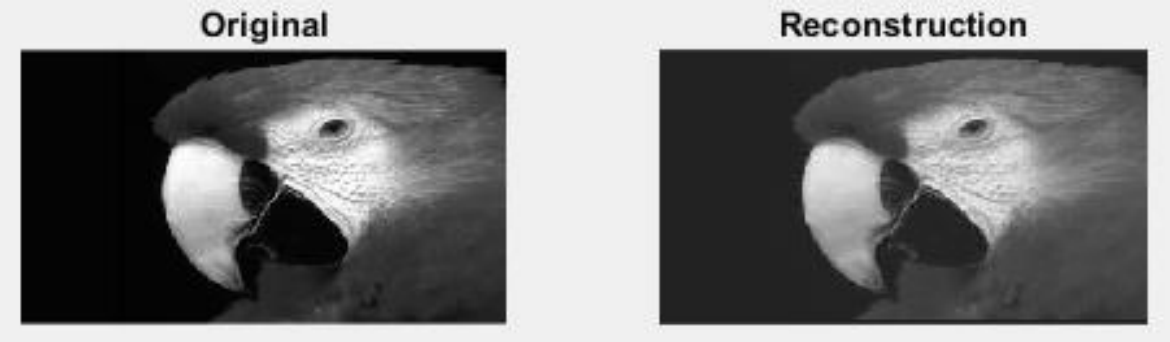


Η άγνωστη γάτα εμφανίζει πιο pixelated υφή, καθώς υπάρχει έλλειψη ατόμων που να ταιριάζουν ακριβώς στα χαρακτηριστικά.

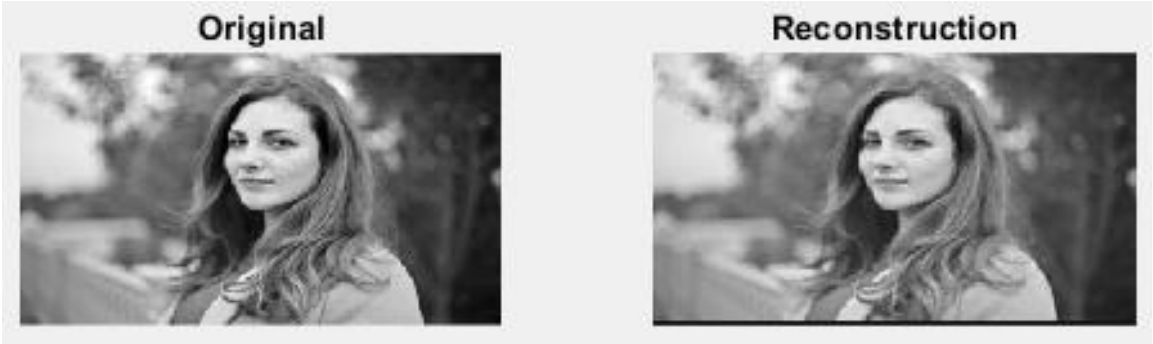
Το γρασίδι, οι φυλλωσιές και οι λεπτομέρειες φόντου ανακατασκευάζονται ικανοποιητικά σε γενικές γραμμές, αλλά χάνεται ποιότητα στις υψηλές συχνότητες. Η εικόνα μοιάζει λίγο πιο ομαλή σε σχέση με την αρχική, γεγονός που αποδεικνύει ότι είχαμε απώλεια υψηλών συχνοτήτων, χωρίς αυτή να είναι σε επίπεδο που έχει χαθεί όλη η λεπτομέρεια της εικόνας.

Οι ευθείες γραμμές, οι καθαρές αντιθέσεις σπιτιού–νερού και το επίπεδο φόντο αναπαράγονται καλά, ωστόσο το νερό και το σύννεφο γίνονται πιο ομαλά. Το συνολικό αποτέλεσμα είναι καλό, αλλά χωρίς την απόλυτη ακρίβεια και λεπτομέρεια του πρωτοτύπου.

Εντυπωσιακά, το πρόσωπο του Keanu δείχνει σχεδόν αναλλοίωτο, αφού οι σκιές στο πρόσωπο και τα χαρακτηριστικά (μάτια, σαγόνι) διατηρούνται με εξαιρετική πιστότητα. Μάλιστα, λόγο της αυξημένης φωτεινότητας μπορούμε πλέον να διακρίνουμε όλο το πρόσωπο που δεν ήταν αρχικά εμφανές.



Τα πολύπλοκα μοτίβα των φτερών εμφανίζουν κάποιες ατέλειες. Οι λεπτομέρειες γύρω από το ράμφος και τα στρώματα του φτερώματος γίνονται λιγότερο έντονα. Ωστόσο, η γενική μορφή και η αντίθεση διατηρούνται.



Το πρόσωπο της γυναίκας διατηρεί το βασικό περίγραμμα και τις μεγάλες περιοχές φωτισμού, αλλά η υφή του δέρματος γίνεται πιο λεία και λιγότερο ευκρινείς.

Γενικά, μπορούμε να δούμε ότι σαφώς υπάρχει απώλεια στην ποιότητα της ανακατασκευής. Ένα ξεκάθαρο μοτίβο είναι ότι έχουμε χάσει υψηλές συχνότητες στις εικόνες, αλλά όχι σε μεγάλο επίπεδο όπως π.χ να χρησιμοποιήσουμε φίλτρο gauss. Το επίπεδο ποιότητας είναι ικανοποιητικό αν σκεφτούμε και τον χρόνο που χρειαστήκαμε για την εκτέλεση του προγράμματος, ο οποίος ήταν λίγο περισσότερο από 3 λεπτά σε επεξεργαστή 10 ετιας. Ίσως με παραπάνω επεξεργασία να έχουμε καλύτερη ανακατασκευή, αν σκεφτούμε ότι έχουμε κρατήσει τις χαμηλές συχνότητες, αφού ξέρουμε ότι η ενέργεια είναι μαζεμένη σε αυτές.

Python Bonus 1

Υλοποιούμε το ίδιο ακριβώς σύστημα σε Python. Συγκεκριμένα, αυτή την φορά τρέχουμε των κώδικα στο google collab και άρα υπάρχει η δυνατότητα να αλλάξουμε τις παραμέτρους ώστε να έχουμε καλύτερο αποτέλεσμα. Κάνουμε την εκπαίδευση και βλέπουμε ότι ο χρόνος που χρειάστηκε είναι τεράστιος σε σχέση με το αρχικό μας σύστημα παρόλο που έχουμε στην διάθεση μας μια πολύ καλή κάρτα γραφικών (2+ ωρες).

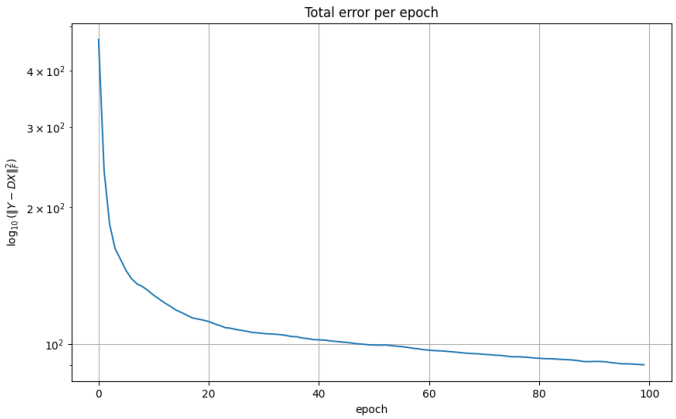
```
GPU: Tesla T4
Total memory: 15.83 GB
Memory allocated: 0.02 GB
Memory after clearing cache: 0.02 GB
D is on GPU: True
X is on GPU: True
Y is on GPU: True

Starting full training with 100 epochs...
Memory after moving tensors: 0.023 GB
Training epochs: 100% ████████████████████ 100/100 [2:09:17<00:00, 77.42s/it]
```

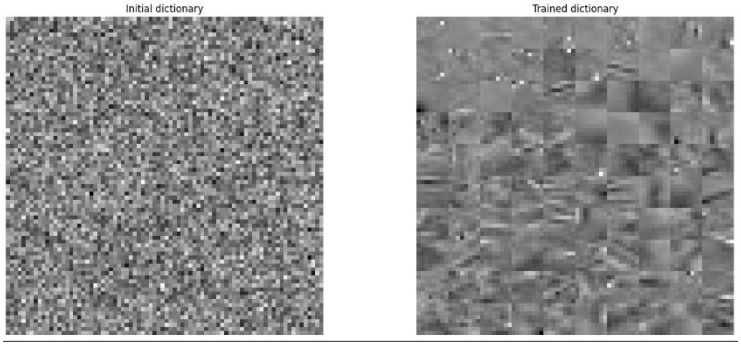
Το τελικό σφάλμα είναι υψηλότερο αυτή την φορά (93 ενώ σε matlab είχαμε 53).

```
Updating dictionary...
MSE: 89.934032
GPU memory after epoch 100: 0.025 GB
Total time: 7757.0 sec
```

Επιπλέον, η καμπύλη σφάλματος έχει αλλάξει σημαντικά. Βλέπουμε ότι περίπου στις 40 εποχές έχουμε πέτυχει σύγκλιση και δεν έχουμε πλέον τα μικρά «κύματα» σε τόσο μεγάλο βαθμό. Αν κοιτάξουμε προσεκτικά βλέπουμε ότι και πάλι υπάρχει ελάχιστη αύξηση του σφάλματος για τον λόγο που έχουμε εξηγήσει παραπάνω, αλλά σε πολύ μικρότερη κλίμακα.



Όσον αφορά το λεξικό μας, ισχύουν οι ιδίες παρατηρήσεις που έχουμε κάνει παραπάνω.



Τέλος, για την ανακατασκευή βλέπουμε ότι θεωρητικά έχουμε πετύχει μικρότερο σφάλμα αλλά πρακτικά ισχύουν οι ίδιες παρατηρήσεις που έχουμε δει. Παραθέτουμε 2 εικόνες ως παράδειγμα.

Denoise-Inpaint

Διαδικασία

1)

Denoising:

Στο τμήμα της αποθορυβοποίησης, το σύστημα ξεκινά με τη φόρτωση του προεκπαιδευμένου λεξικού D διαστάσεων 64×100 του προηγούμενου ερωτήματος. Για κάθε εικόνα test, πρώτα την μετατρέπουμε σε grayscale και την κάνουμε scale κατά 0.25 ώστε να μειώσουμε το μέγεθος της προς επεξεργασία. Η επιλογή μη επικαλυπτόμενων patches (stride = μέγεθος patch = 8) απλοποιεί τη διαδικασία συλλογής και reconstruction, καθώς κάθε pixel ανήκει ακριβώς σε ένα patch.

Στην συνέχεια, για κάθε επίπεδο SNR (0, 20, 50, 100 dB) υπολογίζεται η μέση ισχύς του σήματος και στη συνέχεια η ισχύς του θορύβου. Προστίθεται λευκός Gaussian θόρυβος με τυπική απόκλιση sqrt(Pnoise), παράγοντας την «θορυβώδη» εικόνα. Κάθε 8×8 patch της εικόνας με θόρυβο εξάγεται ως διάνυσμα v διαστάσεων 64×1 και κανονικοποιείται μέσω της L_2-νόρμας. Στη συνέχεια, καλείται ο αλγόριθμος GenOMP και σε κάθε επανάληψη επιλέγεται το άτομο του λεξικού που ελαχιστοποιεί το υπόλοιπο. Αυτό γίνεται μέχρι να επιτευχθεί ο μέγιστος αριθμός μη μηδενικών συντελεστών T0 = 10 ή το μέτρο του υπολοίπου να πέσει κάτω από το όριο err = 1e-4.

Προσθέτοντας, η ανακατασκευή κάθε patch γίνεται πολλαπλασιάζοντας τον συνδυασμό των ατόμων (Dx) με την αρχική νόρμα, αποκαθιστώντας την κλίμακα του patch. Αυτό το ανακτημένο patch τοποθετείται ξανά στη σωστή θέση στο πλαίσιο της εικόνας και η διαδικασία επαναλαμβάνεται για όλα τα patches. Καθώς δεν χρησιμοποιούνται επικαλυπτόμενα patches, δεν απαιτείται συνένωση με ζώνες επικάλυψης.

Τέλος, αφού ολοκληρωθεί η κατασκευή όλων των patches, προκύπτει η τελική αποθορυβοποιημένη εικόνα Irec. Για κάθε επίπεδο θορύβου και κάθε εικόνα, υπολογίζεται το MSE.

Inpainting:

Στο τμήμα inpainting, η διαδικασία ξεκινά όπως στην αποθορυβοποίηση. Κάθε εικόνα μετατρέπεται σε grayscale, κλιμακώνεται κατά 0.25 για μείωση του μεγέθους και χωρίζεται σε μη επικαλυπτόμενα patches 8×8. Η βασική διαφορά είναι ότι εδώ εισάγουμε τόσο θόρυβο όσο και τυχαία gaps στο 50% των pixel,

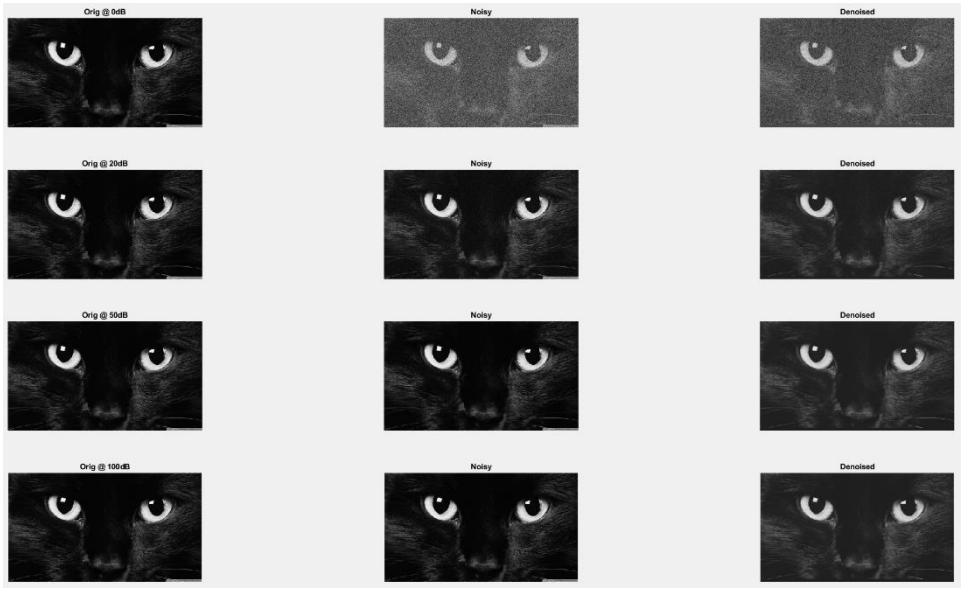
ώστε να δημιουργήσουμε μια «τεχνητή» εικόνα με κενά που πρέπει να συμπληρωθούν.

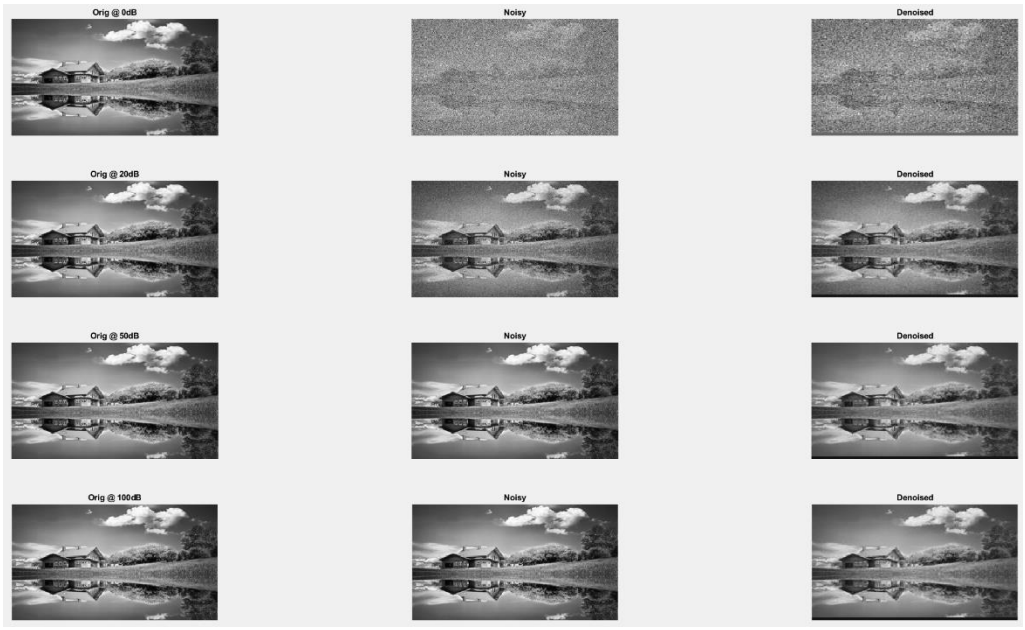
Για κάθε επίπεδο SNR, υπολογίζεται όπως και πριν η ισχύς του θορύβου και προστίθεται λευκός Gaussian θόρυβος στην εικόνα. Στη συνέχεια δημιουργείται μάσκα τυχαίων θέσεων (binary mask) όπου περίπου το 50% των στοιχείων είναι μηδέν (missing) και το υπόλοιπο 50% έχουν τιμή ένα. Στα pixel όπου η μάσκα είναι μηδενικά, η εικόνα έχει τιμή μηδέν.

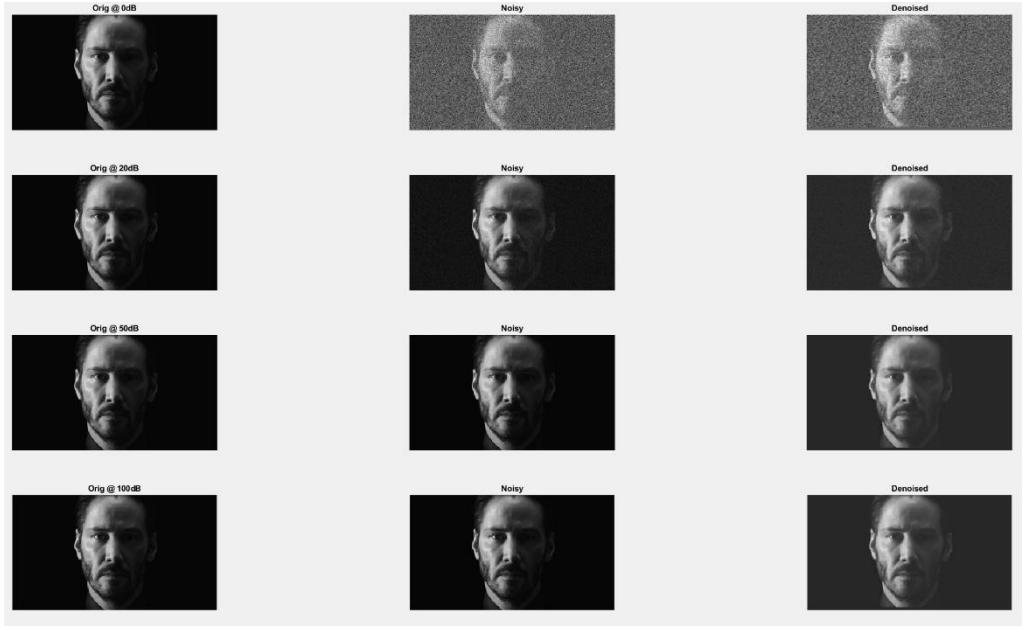
Επιπλέον, η κεντρική ιδέα είναι ότι, αντί να χρησιμοποιήσουμε ολόκληρο το patch για την εύρεση των συντελεστών x, λαμβάνουμε υπόψη μόνο τα παρατηρούμενα pixel. Για κάθε patch, κατασκευάζεται το διάνυσμα vec με τα 64 στοιχεία και το διάνυσμα msk που δείχνει ποια από αυτά είναι διαθέσιμα. Αν όλα τα στοιχεία είναι κενά (msk=0), απλώς διατηρούμε το patch όπως είναι (μηδενικό). Διαφορετικά, εξάγουμε το υπο-διάνυσμα ysub και εφαρμόζουμε τον OMP στο μερικώς παρατηρούμενο λεξικό D(msk,:), χρησιμοποιώντας την κανονικοποίηση ysub/norm(ysub).

Τέλος, η επανακατασκευή του πλήρους patch γίνεται πολλαπλασιάζοντας Dx με την αρχική νόρμα, αποκαθιστώντας την κλίμακα και στη συνέχεια αναδιατάσσοντας το αποτέλεσμα σε 8×8. Το ανακτημένο patch τοποθετείται στη θέση του στην εικόνα, συμπληρώνοντας τόσο τα κενά όσο και αποθορυβοποιώντας τα παρατηρούμενα pixel ταυτόχρονα. Μετά την επεξεργασία όλων των patches, προκύπτει η πλήρως ανακατασκευασμένη εικόνα χωρίς κενά.

2)



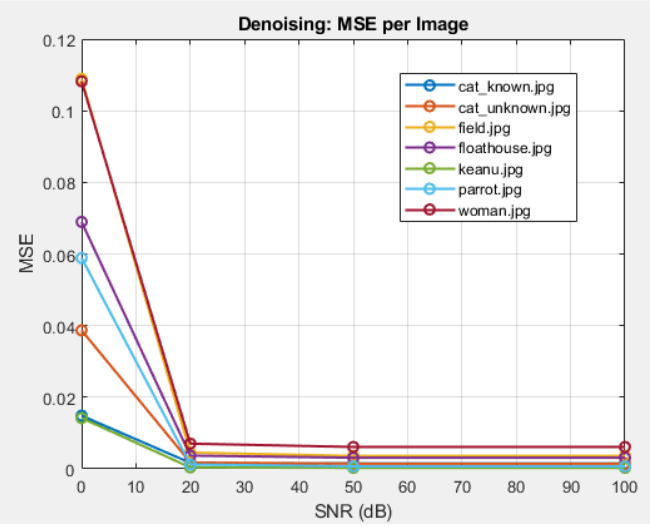

Orig @ 0dB
Noisy
Denoised
Orig @ 20dB
Noisy
Denoised
Orig @ 50dB
Noisy
Denoised
Orig @ 100dB
Noisy
Denoised


Orig @ 0dB
Noisy
Denoised
Orig @ 20dB
Noisy
Denoised
Orig @ 50dB
Noisy
Denoised
Orig @ 100dB
Noisy
Denoised

Αρχικά, παρατηρούμε ότι η ποιότητα ανακατασκευής βελτιώνεται σταδιακά καθώς αυξάνεται το SNR από 0 dB στα 100 dB. Στα 0 dB το επίπεδο θορύβου είναι τόσο υψηλό που πολλές λεπτομέρειες εξαφανίζονται και παρότι ο αλγόριθμος αφαιρεί σημαντικό μέρος του θορύβου, η εικόνα παραμένει δυσδιάκριτη. Στα 20 dB διακρίνεται ήδη ένα σημαντικό «ξεθόλωμα» των βασικών σχημάτων, ενώ στα 50 dB τα περισσότερα χαρακτηριστικά, όπως οι ακμές και οι μεταβάσεις φωτεινότητας, επανακτώνται με καλή πιστότητα. Τέλος, στα 100 dB το αποτέλεσμα προσεγγίζει σε μεγάλο βαθμό την αρχική καθαρή εικόνα, με ελάχιστο ορατό θόρυβο και ευκρινείς λεπτομέρειες.

Γενικότερα, σε όλες τις εικόνες στα 50 db και άνω ο θόρυβος πρακτικά δεν επηρεάζει οπτικά το αποτέλεσμα. Επιπλέον, σε εικόνες με σκούρο background ακόμη και στα 20 db ο θόρυβος δεν χαλάει σε μεγάλο επίπεδο το τελικό αποτέλεσμα, ενώ αντιθέτως σε εικόνες με ανοιχτό background υπάρχει σαφή υποβάθμιση.

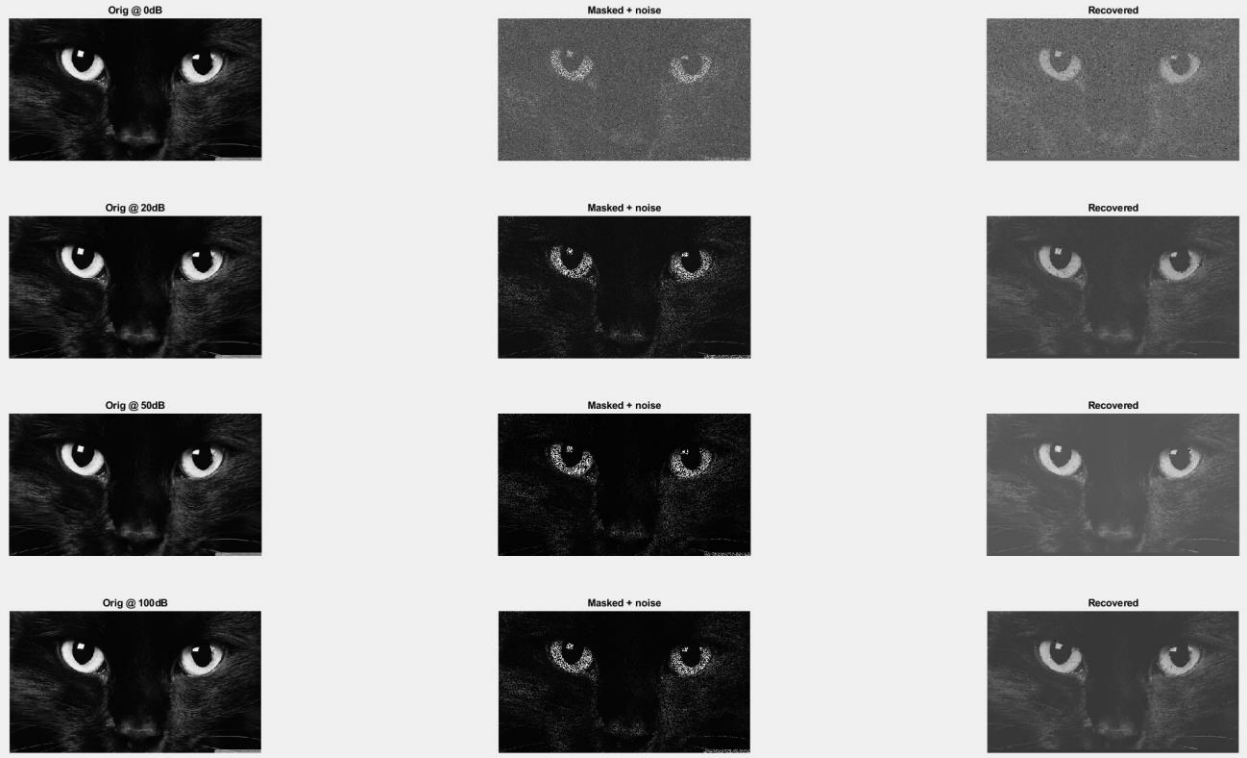
3)

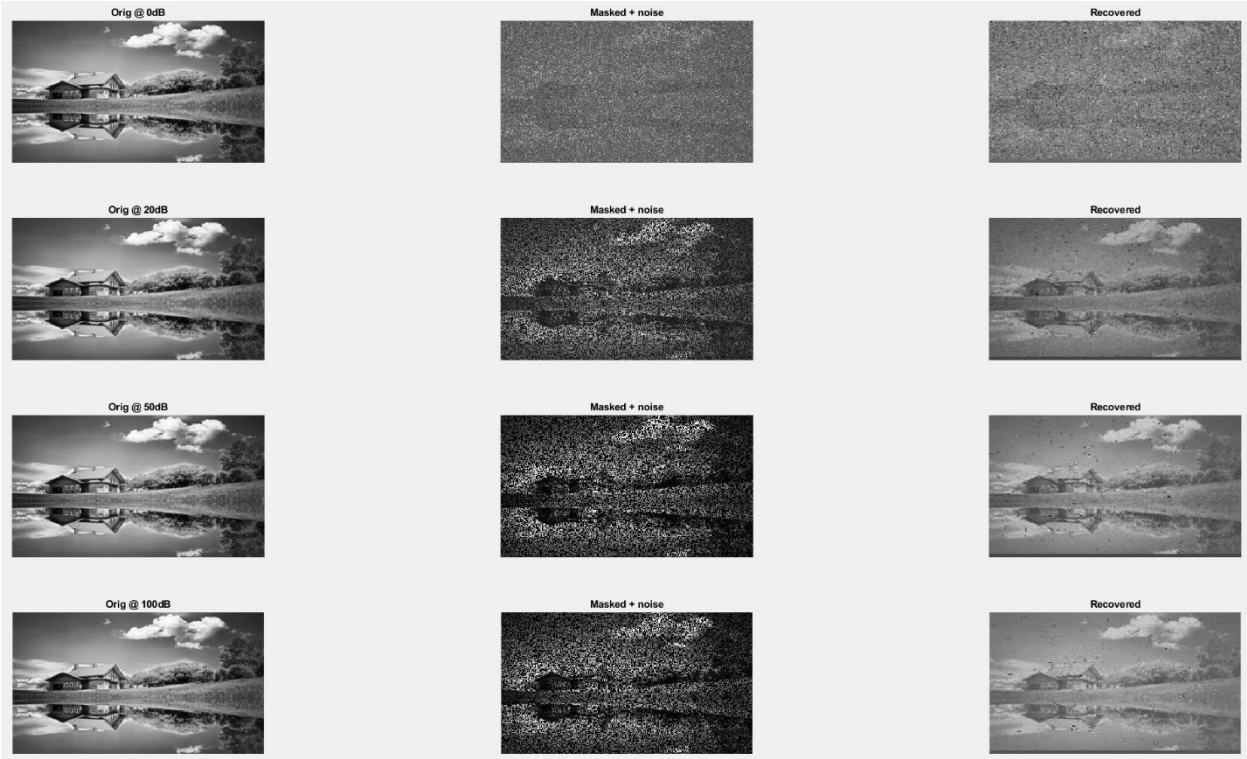


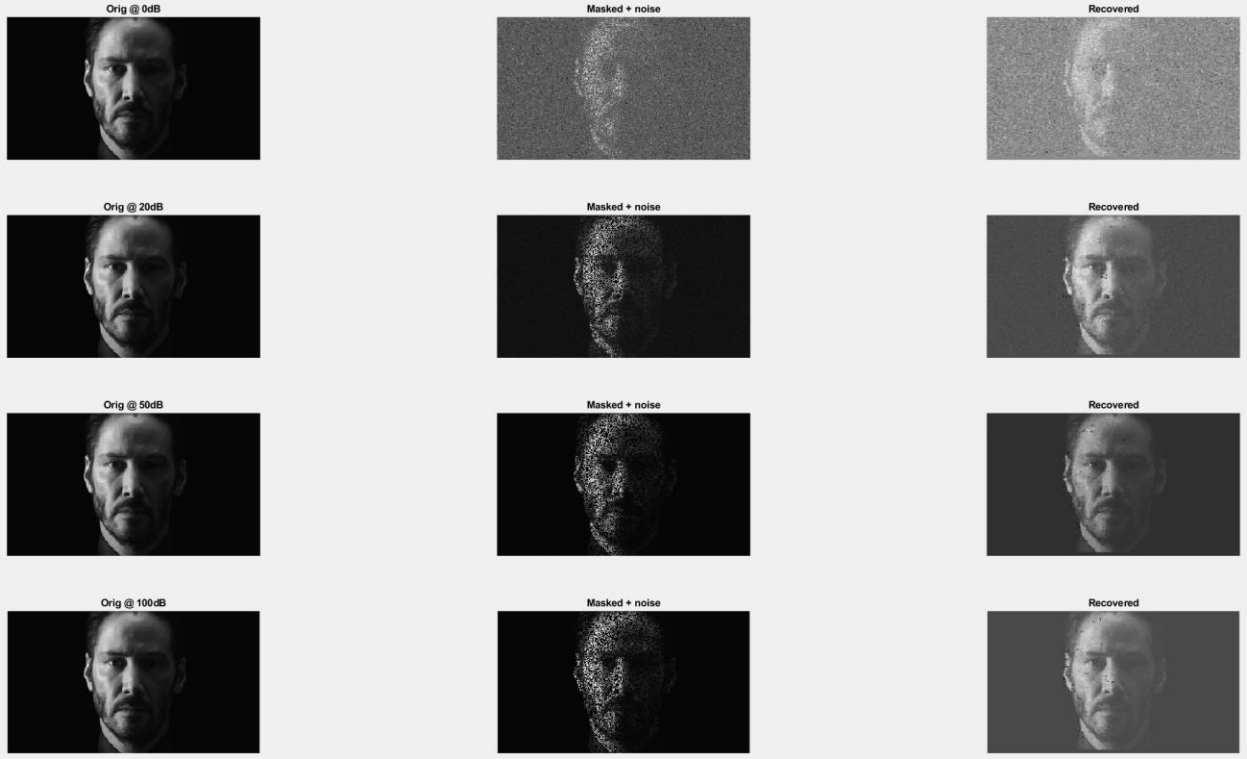
Αναλύοντας τις καμπύλες, διακρίνουμε τρία βασικά χαρακτηριστικά. Πρώτον, σε χαμηλά επίπεδα SNR (0 dB), το σφάλμα είναι ιδιαίτερα υψηλό (γύρω στο 0.1–0.11), καθώς το θορυβώδες σήμα επικαλύπτει σχεδόν πλήρως τις λεπτομέρειες της εικόνας. Καθώς ανεβαίνουμε στο 20 dB, το σφάλμα υποχωρεί απότομα σε πολύ χαμηλά επίπεδα (κάτω από 0.01 για όλες τις εικόνες.

Δεύτερον, παρατηρείται ότι μετά τα 20 dB η πτώση του σφάλματος γίνεται πολύ πιο απότομα και συγκλίνει μεταξύ 50 dB και 100 dB. Αυτό συμβαίνει επειδή, σε αυτά τα επίπεδα θορύβου, ο αλγόριθμος έχει ήδη ανακτήσει την εικόνα. Όσον αφορά την διαφορά σφάλματος ανάμεσα στις εικόνες, έχει εξηγηθεί παραπάνω σε προηγούμενα ερωτήματα.

4)


Orig @ 0dB
Masked + noise
Recovered
Orig @ 20dB
Masked + noise
Recovered
Orig @ 50dB
Masked + noise
Recovered
Orig @ 100dB
Masked + noise
Recovered


Orig @ 0dB
Masked + noise
Recovered
Orig @ 20dB
Masked + noise
Recovered
Orig @ 50dB
Masked + noise
Recovered
Orig @ 100dB
Masked + noise
Recovered


Orig @ 0dB
Masked + noise
Recovered
Orig @ 20dB
Masked + noise
Recovered
Orig @ 50dB
Masked + noise
Recovered
Orig @ 100dB
Masked + noise
Recovered

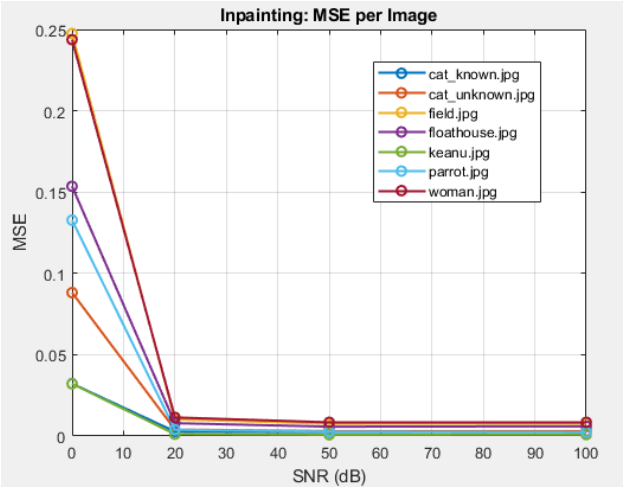

Orig @ 0dB
Masked + noise
Recovered
Orig @ 20dB
Masked + noise
Recovered
Orig @ 50dB
Masked + noise
Recovered
Orig @ 100dB
Masked + noise
Recovered

Βλέπουμε ότι, σε όλες τις εικόνες, ο αλγόριθμος παρουσιάζει σαφή βελτίωση όσο αυξάνεται το SNR, τόσο στην καταστολή του θορύβου όσο και στην ακρίβεια ανακατασκευής. Στα 0 dB, όπου ο θόρυβος έχει τη μέγιστη ισχύ σε σχέση με το σήμα, οι λεπτομέρειες χάνονται σχεδόν πλήρως, καθώς ο αλγόριθμος καλείται να συμπληρώσει μεγάλα κενά με ελάχιστες έγκυρες παρατηρήσεις.

Στα 20 dB παρατηρείται όμως σημαντική ανάκτηση βασικών δομών. Τα τυχαία κενά (50% των pixel) εξακολουθούν να εμφανίζονται ως σκοτεινές ή φωτεινές τρύπες στη «Masked + noise», αλλά η «Recovered» εικόνα ανακτά σαφείς περιγράμματα π.χ τα μάτια της γάτας φαίνονται, οι γραμμές του τοπίου γίνονται εμφανείς και τα περίγραμμα του προσώπου επανέρχονται. Παρότι πολλά εσωτερικά pixels παραμένουν ελαφρώς θολά, οι κύριες δομές γίνονται αναγνωρίσιμες, δείχνοντας ότι το λεξικό μπορεί να γεμίσει αξιόπιστα μεγάλα κενά μόλις ο θόρυβος υποχωρήσει κάτω από κάποιο όριο.

Τέλος, στα μεγαλύτερα db αυξάνεται η ποιότητα των εικόνων, χωρίς όμως το τελικό αποτέλεσμα να είναι τέλειο.

5)



Τα αποτελέσματα είναι ίδια με αυτά του ερωτήματος 3.

Python Bonus 2

Δημιουργούμε το σύστημα στο google collab και από τις εικόνες βλέπουμε ότι τα αποτελέσματα είναι ιδία με αυτά της matlab.

MSE Denoising (rows=images / cols=SNR[dB]):

	dB_0	dB_20	dB_50	dB_100
woman.jpg	0.123673	0.007347	0.006273	0.006269
cat_unknown.jpg	0.045222	0.002490	0.002110	0.002109
keanu.jpg	0.016571	0.000675	0.000521	0.000521
floathouse.jpg	0.079283	0.004855	0.004193	0.004191
cat_known.jpg	0.017656	0.002151	0.002001	0.002000
field.jpg	0.127166	0.006528	0.005459	0.005455
parrot.jpg	0.066720	0.001503	0.000899	0.000897

MSE Inpainting (rows=images / cols=SNR[dB]):

	dB_0	dB_20	dB_50	dB_100
woman.jpg	0.228478	0.010312	0.008114	0.008472
cat_unknown.jpg	0.084888	0.005246	0.004532	0.004468
keanu.jpg	0.031127	0.001816	0.001479	0.001420
floathouse.jpg	0.146915	0.009747	0.008157	0.008389
cat_known.jpg	0.032554	0.004053	0.003880	0.003694
field.jpg	0.243973	0.014156	0.011461	0.011318
parrot.jpg	0.126257	0.003668	0.002583	0.002963