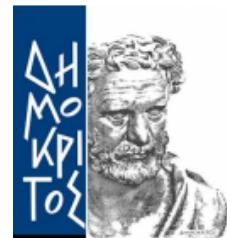


ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ  
ΠΟΛΥΤΕΧΝΕΙΟ



ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΕΡΕΥΝΑΣ  
ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ  
“ΔΗΜΟΚΡΙΤΟΣ”



Σχολή Ηλεκτρολόγων Μηχανικών  
& Μηχανικών Υπολογιστών

Ινστιτούτο Νανοεπιστήμης &  
Νανοτεχνολογίας  
Εργαστήριο Στατιστικής  
Μηχανικής & Πολύπλοκων  
Δυναμικών Συστημάτων

---

**Ανάλυση εικόνων  
Μαγνητικής Τομογραφίας με  
χρήση Συνελικτικών  
Νευρωνικών Δικτύων και  
αλγορίθμων Οπτικοποίησης**

---

Παναγιωταράς Ηλίας

9 Ιανουαρίου 2020

## Περίληψη

Το παρόν αποτελεί αναφορά της εργασίας που πραγματοποιήθηκε στο Εργαστήριο Στατιστικής Μηχανικής και Πολύπλοκων Δυναμικών Συστημάτων του Ινστιτούτου Νανοεπιστήμης και Νανοτεχνολογίας του ΕΚΕΦΕ «Δημόκριτος» στα πλαίσια του προγράμματος πρακτικής άσκησης της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Σε αυτή την εργασία εκπαιδεύσαμε ένα συνελικτικό τρισδιάτατο νευρωνικό δίκτυο να αναγνωρίζει δομικές διαφορές που σχετίζονται με τη νόσο του Αλτσχάιμερ βασιζόμενο σε εικόνες δομικού MRI εγκεφάλων. Για την διαγνωστική ταξινόμηση έγινε χρήση δύο σετ δεδομένων, το ένα με 180 3D εικόνες των 3 Tesla (T) και το άλλο με 969 3D εικόνες των 1.5 Tesla. Το δίκτυο πέτυχε ακρίβεια ταξινόμησης 69% στα δεδομένα των 3T και ακρίβεια 79% στα δεδομένα των 1.5T. Έπειτα, εφαρμόσαμε τέσσερις αλγορίθμους οπτικοποίησης, που εξηγούν τις αποφάσεις ταξινόμησης του δικτύου, επισημαίνοντας τις σχετικές περιοχές στις εικόνες εισόδου. Συγκρίναμε τις μεθόδους ποιοτικά και ποσοτικά, και βρέθηκε ότι το δίκτυο εστιάζει σε εγκεφαλικές περιοχές οι οποίες είναι γνωστό από τη βιβλιογραφία πως σχετίζονται άμεσα με την νόσο, όπως η άνω και η μέση κροταφική έλικα. Παράλληλα, η κατανομή της σχετικότητας φαίνεται να ποικίλλει ανάμεσα στους ασθενείς, με ορισμένους να εμφανίζουν περισσότερες διαφοροποιήσεις σε περιοχές του κροταφικού λοβού, ενώ σε άλλους διαφοροποιούνται πιο φλοιϊκές περιοχές. Τέλος, δείχνουμε πως η εφαρμογή αλγορίθμων οπτικοποίησης σε νευρωνικά δίκτυα αποτελεί ένα σημαντικό βήμα στην αύξηση της διαγνωστικής τους αξίας, καθώς και της κλινικής εμπιστοσύνης στα υπολογιστικά συστήματα υποστήριξης λήψης αποφάσεων.

## Ευχαριστίες

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους επιβλέποντες της πρακτικής μου άσκησης, Αστέρω Προβατά, Ερευνήτρια Α, στο ΕΚΕΦΕ ”Δημόκριτος” και επιστημονική επόπτη της εργασίας, και Ιωάννη Γκόνο, Επίκουρο Καθηγητή στο Εθνικό Μετσόβιο Πολυτεχνείο. Η ευκαιρία να δουλέψω μαζί τους ήταν μοναδική και οι εμπειρίες που αποκόμισα πολύτιμες. Θέλω να τους ευχαριστήσω για τη συνολική τους στάση απέναντι μου, την επιστημονική τους καθοδήγηση και την εμπιστοσύνη που μου έδειξαν. Ακόμα, θέλω να ευχαριστήσω την υποψήφια διδάκτορα Νεφέλη-Δήμητρα Τσίγκρη, τον διπλωματούχο Ηλεκτρολόγο Μηχανικό και Μηχανικό Υπολογιστών Κουλιεράκη Ιωάννη, καθώς και τους συναδέλφους μου Φερίκογλου Άγγελο, Διολέτη Τλια και Χαρδούβελη Ορέστη. Η συνεργασία μας ήταν αρμονική, παραγωγική και συντέλεσε σε ένα δημιουργικό και ευχάριστο δίμηνο. Όλα όσα έμαθα κατά τη διάρκεια της πρακτικής μου άσκησης, τεχνικά και μη, τα οφείλω στους παραπάνω, γι' αυτό και νιώθω τυχερός που ήμουν μέρος αυτού του συνόλου. Τέλος, θα ήθελα να ευχαριστήσω το Εθνικό Δίκτο Υποδομών Τεχνολογίας και Έρευνας (ΕΔΥΤΕ) ([hpc.grnet.gr/](http://hpc.grnet.gr/)) για την παραχώρηση υπολογιστικού χρόνου στο εθνικό υπερ-υπολογιστικό σύστημα ARIS στα πλαίσια του έργου pr007011, όπου πραγματοποιήθηκε ένα μεγάλο μέρος των προσομοιώσεων της εργασίας.

# **Περιεχόμενα**

<b>1</b>	<b>Εισαγωγή</b>	<b>4</b>
1.1	Νόσος Αλτσχάιμερ . . . . .	4
1.2	Συνελικτικά Νευρωνικά Δίκτυα . . . . .	4
1.3	Οπτικοποίηση Νευρωνικών Δικτύων . . . . .	6
1.4	Σχετικές Εργασίες . . . . .	6
<b>2</b>	<b>Δεδομένα</b>	<b>8</b>
<b>3</b>	<b>Τλοποίηση</b>	<b>9</b>
3.1	Επεξεργασία Δεδομένων . . . . .	10
3.2	Μοντέλο . . . . .	13
3.3	Οπτικοποίηση . . . . .	15
<b>4</b>	<b>Αποτελέσματα</b>	<b>16</b>
4.1	Ταξινόμηση . . . . .	16
4.2	Σχετικές εγκεφαλικές περιοχές . . . . .	18
4.3	Διαφορές μεταξύ των μεθόδων οπτικοποίησης . . . . .	19
<b>5</b>	<b>Συμπεράσματα</b>	<b>20</b>
<b>A</b>	<b>Παράρτημα</b>	<b>23</b>

# 1 Εισαγωγή

Η βαθιά μάθηση, μία από τις πιο μοντέρνες προσεγγίσεις στην μηχανική μάθηση, έχει δείξει σημαντικά καλύτερες επιδόσεις στην αναγνώριση μικροδομών σε σύνθετα δεδομένα υψηλών διαστάσεων έναντι των παραδοσιακών μεθόδων υπολογιστικής μάθησης και ιδιαίτερα στο πεδίο της όρασης υπολογιστών. Η εφαρμογή της βαθιάς μάθησης στην έγκαιρη διάγνωση και αυτόματη ταξινόμηση της νόσου του Αλτσχάιμερ έχει, πρόσφατα, κεντρίσει το ενδιαφέρον πολλών ερευνητών, καθώς η ταχεία εξέλιξη των νευροαπεικονιστικών τεχνικών έχει παράξει μεγάλης κλίμακας δεδομένα νευροαπεικόνισης από ποικίλα απεικονιστικά συστήματα [12, 11, 28]. Σημειώνεται πως, ενώ στις επόμενες ενότητες θα γίνει αναφορά στις βασικές μεθόδους που χρησιμοποιήθηκαν, θωρεύται δεδομένη η γνώση των θεμελιώδων εννοιών των πεδίων της τεχνητής νοημοσύνης και συγκεκριμένα της βαθιάς μηχανικής μάθησης.

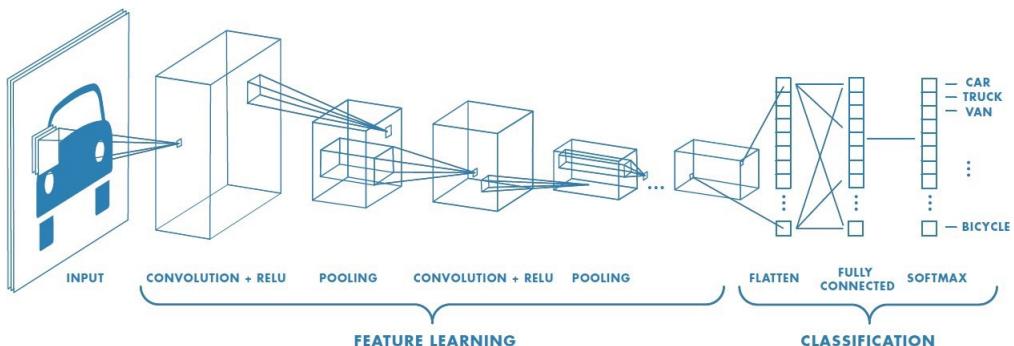
## 1.1 Νόσος Αλτσχάιμερ

Η νόσος του Αλτσχάιμερ (Alzheimer's disease - AD) είναι η πιο επικίνδυνη μορφή άνοιας. Χαρακτηρίζεται από απώλεια μνήμης και άλλων γνωσιακών ικανοτήτων σε βαθμό που επηρεάζεται άμεσα η καθημερινή ζωή των ασθενών[29]. Για ένα μη αμελητέο χρονικό διάστημα πρωτού τα προβλήματα μνήμης εμφανιστούν, μικροσκοπικές αλλαγές που σχετίζονται με τα εγκεφαλικά κύτταρα λαμβάνουν χώρα και σταδιακά εξελίσσονται. Η νευροεκφύλιση είναι το κύριο χαρακτηριστικό της νόσου, ξεκινώντας από τον χροταφικό λοβό, ενώ με την πάροδο του χρόνου εξαπλώνεται σε μία πληθώρα περιοχών του εγκεφάλου. Ωστόσο, δεδομένου ότι η συντριπτική πλειοψηφία των εγκεφάλων ηλικιωμένων ατόμων εμφανίζουν συμπτώματα παρόμοιας ατροφίας με εκείνα της νόσου, η διάκριση ενός υγιούς ηλικιωμένου ατόμου από έναν ασθενή με Αλτσχάιμερ καθίσταται ένα πολύ απαιτητικό και δύσκολο έργο[28].

## 1.2 Συνελικτικά Νευρωνικά Δίκτυα

Στο παραπάνω πλαίσιο, η προσφορά των μοντέλων μηχανικής μάθησης στην αναγνώριση αλλαγών σε μικροδομές είναι μεγάλη. Τα μοντέλα που πετυχαίνουν τις υψηλότερες επιδόσεις για προβλήματα ταξινόμησης εικόνων (image classification problems) είναι τα συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks - CNNs), τα οποία έχουν πρόσφατα αρχίσει να εφαρμόζονται σε ιατρικά δεδομένα απεικόνισης για διάφορες περιπτώσεις χρήσης, συμπεριλαμβανομένου και της διάγνωσης του Αλτσχάιμερ. Η βασική ιδέα πίσω από τα CNNs είναι εμπνευσμένη από τον μηχανισμό των υποδεκτικών πεδίων (receptive fields) στον πρωτεύοντα οπτικό φλοιό. Ο μηχανισμός αυτός μπο-

ρεί να περιγραφεί από την εφαρμογή τοπικών συνελικτικών φίλτρων και λειτουργιών συγκέντρωσης (pooling operations) με στόχο την εξαγωγή τοπικής πληροφορίας. Στο Σχήμα 1 φαίνεται η τυπική δομή ενός τέτοιου μοντέλου. Με βάση αυτή, μπορεί να προκύψει μία πληθώρα από όμοια μοντέλα, τα οποία χαρακτηρίζονται από τα ίδια θεμελιώδη επίπεδα (layers) αλλά σε διαφορετικές διατάξεις και με διαφορετικές παραμέτρους. Σε αντίθεση με τις παραδοσιακές προσεγγίσεις μηχανικής μάθησης, τα συνελικτικά δίκτυα δεν βασίζονται σε χαρακτηριστικά των δεδομένων που έχουν εξαχθεί από τον άνθρωπο για την ταξινόμησή τους, αλλά βρίσκουν ουσιώδεις αναπαραστάσεις των δεδομένων κατά τη διάρκεια της εκπαίδευσης. Σημειώνεται πως με τον όρο εκπαίδευση, εννοούμε τη διαδικασία κατά την οποία ένα νευρωνικό δίκτυο εξάγει τα απαραίτητα χαρακτηριστικά από τα δεδομένα, τα χρησιμοποιεί για να ταξινομήσει την κάθε είσοδο σε μία από τις διαφορετικές κλάσεις, συγχρίνει την πρόβλεψή του με το σωστό αποτέλεσμα σε κάθε βήμα και διορθώνει τις εσωτερικές του παραμέτρους ανάλογα, με σκοπό να βελτιώσει την επίδοσή του. Αυτό βέβαια απαιτεί δεδομένα τα οποία εκτός από την ωμή πληροφορία που περιλαμβάνουν (στην περίπτωση της νευροαπεικόνισης έχουμε τα αρχεία εικόνας), χρειάζεται να συνοδεύονται και από κατάλληλες ετικέτες (labels), ώστε το δίκτυο σε κάθε βήμα να μπορεί να συγχρίνει την πρόβλεψή του με την αντίστοιχη ετικέτα της κάθε εισόδου και να βελτιώσει την επίδοσή του.



Σχήμα 1: Τυπική δομή ενός συνελικτικού νευρωνικού δικτύου. Η είσοδος του δικτύου μπορεί να είναι μία εικόνα δεδομένης ανάλυσης, ενώ η έξοδος το αποτέλεσμα της ταξινόμησης της εισόδου σε διαφορετικές κατηγορίες. Το δίκτυο τροφοδοτείται από δεδομένα και σταδιακά "μαθαίνει" να διαχρίνει τις διαφορές των δεδομένων κατηγοριών, τροποποιώντας τις εσωτερικές παραμέτρους του κάθε επιπέδου. Κάθε επίπεδο εξάγει από την έξοδο του προηγούμενου επιπέδου την πληροφορία που θεωρεί πιο σημαντική, υλοποιεί, δηλαδή, αυτό που ονομάζουμε εξαγωγή χαρακτηριστικών. (towardsdatascience.com)

### 1.3 Οπτικοποίηση Νευρωνικών Δικτύων

Παρότι τα συνελικτικά νευρωνικά δίκτυα χαρακτηρίζονται από υψηλά ποσοστά ακρίβειας σε προβλήματα ταξινόμησης, είναι αρκετά δύσκολο να οπτικοποιηθούν οι αποφάσεις τους και να ερμηνευτούν κατάλληλα. Με τον όρο αποφάσεις, εννοούμε τα τελικά χαρακτηριστικά που επέλεξαν να εστιάσουν για να διακρίνουν τα δεδομένα στις διαθέσιμες κατηγορίες. Στο δεδομένο πρόβλημα, τα χαρακτηριστικά αυτά θα χαρακτηρίζουν συγκεκριμένες περιοχές του εγκεφάλου, όπως αυτές φαίνονται μέσα από τις εικόνες νευροαπεικόνισης. Ωστόσο, κατα τη διαδικασία λήψης ιατρικών αποφάσεων είναι απαραίτητο να εξηγηθεί η συμπεριφορά του μοντέλου μηχανικής μάθησης, ώστε οι ειδικοί να είναι σε θέση να διασταυρώσουν τις αποφάσεις του δικτύου με την ήδη υπάρχουσα γνώση στη βιβλιογραφία και να επιβεβαιώσουν την διάγνωση του συστήματος. Έχει προταθεί μία πληθώρα από μεθόδους οπτικοποίησης δικτύων, οι οποίες επισημαίνουν περιοχές στις εικόνες εισόδου που έπαιξαν καθοριστικό ρόλο στην λήψη της απόφασης ταξινόμησης [21, 22, 23, 24]. Οι έξοδοι τέτοιων αλγορίθμων, έχουν τη μορφή χαρτών σχετικότητας με κλίμακα τη θερμότητα των εικονιζόμενων χρωμάτων (heatmaps). Οι εν λόγω χάρτες θερμότητας αποτελούν τη βάση τόσο για την κατανόηση όσο και για την ερμηνεία των μοντέλων μηχανικής μάθησης, ιδιαίτερα σε συνεργασία με τους κλινικούς ιατρούς.

Στην παρούσα εργασία υλοποιήσαμε ένα μοντέλο τρισδιάστατου συνελικτικού νευρωνικού δικτύου, το οποίο εκπαιδεύτηκε με δεδομένα μαγνητικής τομογραφίας που προήλθαν από τη διεθνή βάση δεδομένων ADNI. Επειτα, συγκρήθηκαν τέσσερις μέθοδοι οπτικοποίησης, ανάλυση ευαισθησίας (sensitivity analysis), ελεγχόμενη οπισθοδιάδοση (guided backpropagation), έγκλειση (occlusion) και περιοχική έγκλειση (area occlusion). Το δεδομένο πρόβλημα ταξινόμησης χαρακτηρίζεται από δύο κατηγορίες, ασθενείς με τη νόσο του Αλτσχάιμερ (AD patients) και γνωστικά υγιή ηλικιωμένα άτομα (normal elderly controls - NCs).

### 1.4 Σχετικές Εργασίες

Στη διεθνή βιβλιογραφία, υπάρχει ένας μεγάλος αριθμός μοντέλων που έχουν εφαρμοστεί στην αναγνώριση της νόσου του Αλτσχάιμερ. Πολλές από αυτές κάνουν χρήση παραδοσιακών προσεγγίσεων με χαρακτηριστικά που έχουν επιλεχθεί χειροκίνητα. Ωστόσο, οι πιο πρόσφατες έρευνες εστιάζουν αποκλειστικά σε μοντέλα συνελικτικών νευρωνικών δικτύων. Στον Πίνακα 1 φαίνεται μία αναλυτική περιγραφή των σχετικών εργασιών, το οποίο προέρχεται από τη δουλειά των Khvostikov et al. [25]. Τρείς από τις αναφερόμενες εργασίες χρησιμοποιούν μοντέλα και διαδικασίες εκπαίδευσης παρόμοιες με

αυτές της παρούσας εργασίας, δηλαδή κάνουν χρήση τρισδιάστατων συνελικτικών νευρωνικών δικτύων και εικόνες δομικού MRI (structural Magnetic Resonance Imaging - sMRI) με στόχο την ταξινόμηση υγιών ατόμων και ασθενών με AD [26, 14, 8]. Σε αντίθεση με την τωρινή εργασία, οι Payan et al. [8] και Hosseini et al. [26] εφαρμόζουν έναν μη επιβλεπόμενο αυτοκαθικοποιητή (unsupervised autoencoder) πριν το στάδιο της εκμάθησης του δικτύου, ενώ οι Korolev et al. [14] εκπαιδεύουν το μοντέλο τους απευθείας, χρησιμοποιώντας παράλληλα ένα πιο βαθύ νευρωνικό δίκτυο. Η αρχιτεκτονική του συνελικτικού δικτύου της παρούσας εργασίας είναι εν μέρει εμπνευσμένη από το μοντέλο των Khvostikov et al. [25]. Η θεμελιώδης διαφορά είναι το γεγονός ότι στην εργασία τους χρησιμοποιούν μόνο εικόνες του ιπποκάμπου, ενώ στην παρούσα γίνεται χρήση εικόνων από ολόκληρο τον εγκέφαλο.

Πίνακας 1: Αναλυτική σύγκριση διαφορετικών μοντέλων ταξινόμησης. Στον πίνακα αναφέρεται η μεθοδολογία (Methodology) που ακολουθήθηκε (χειροκίνητη εξαγωγή χαρακτηριστικών (Feature-based), αυτόματη εξαγωγή χαρακτηριστικών μέσω νευρωνικού δικτύου (NN-based) ή διάγνωση μέσω ογκομετρικής μορφολογίας (Volumetric)), το απεικονιστικό μηχάνημα από το οποίο προήλθαν τα δεδομένα (Modalities), οι περιοχές του εγκεφάλου οι οποίες χρησιμοποιήθηκαν για την εξαγωγή των χαρακτηριστικών (Content), ο αριθμός εικόνων μαζί με την πηγή προέλευσής τους (Data (size)) και τέλος, η ακρίβεια που επιτεύχθηκε στα διαφορετικά προβλήματα ταξινόμησης (Accuracy) (άτομα με AD/υγιείς, άτομα με AD/άτομα που πάσχουν από ήπια γνωστική δυσλειτουργία-ΗΓΔ (Mild Cognitive Impairment - MCI) και άτομα με MCI/υγιείς).[25]

Algorithm	Methodology	Modalities	Content	Data (size)	Accuracy		
					AD/NC	AD/MCI	MCI/NC
Magnin et al. [7]	Volumetric	sMRI	Full brain	custom (38)	0.945	-	-
Ahmed et al. [10]	Feature-based	sMRI	2 ROIs	ADNI (509)	0.838	0.695	0.621
Ebadie et al. [14]	Feature-based	DTI	Full brain	custom (34)	0.8	0.833	0.7
Lee et al. [15]	Feature-based	DTI	Full brain	LONI (141)	0.977	0.977	-
Lei et al. [16]	Feature-based	sMRI + PET	Full brain	ADNI (398)	0.969	-	0.866
Ahmed et al. [11]	Feature-based	sMRI + DTI	1 ROI	ADNI (203)	0.902	0.766	0.794
Vu et al. [3]	NN-based	sMRI + PET	Full brain	ADNI (203)	0.91	-	-
Payan and Montana [4]	NN-based	sMRI	Full brain	ADNI (2265)	0.993	1	0.942
Glozman and Liba [21]	NN-based	sMRI + PET	Full brain	ADNI (1370)	0.665	-	-
Sarraf et al. [24]	NN-based	sMRI, fMRI	Full brain	ADNI (302)	0.988, 0.999	-	-
Billones et al. [5]	NN-based	sMRI	Full brain	ADNI (900)	0.983	0.939	0.917
Aderghal et al. [28]	NN-based	sMRI	1 ROI	ADNI (815)	0.914	0.695	0.656
Shi et al. [30]	NN-based	sMRI + PET	Full brain	ADNI (202)	0.971	-	0.872
Korolev et al. [31]	NN-based	sMRI	Full brain	ADNI (231)	0.79-0.8	-	-
Suk et al. [32]	NN-based	sMRI	93 ROIs	ADNI (805)	0.903	-	0.742
Suk et al. [32]	NN-based	sMRI	93 ROIs	ADNI (805)	0.903	-	0.742
Luo et al. [18]	NN-based	sMRI	7 ROIs	ADNI (81)	0.83	-	-
Wang et al. [23]	NN-based	sMRI	Full brain	ADNI (629)	-	-	0.906
Li et al. [33]	NN-based	sMRI	1 ROI	ADNI (1776)	0.965	0.67	0.622
Cheng et al. [20]	NN-based	sMRI	27 ROIs	ADNI (1428)	0.872	-	-
Li et al. [34]	NN-based	sMRI	Full brain	ADNI (832)	0.91	0.877	0.855

Τα τελευταία χρόνια, το πεδίο της οπτικοποίησης συνελικτικών νευρωνικών δικτύων έχει κεντρίσει το ενδιαφέρον ενός αυξανόμενου αριθμού ερευνητών, με αποτέλεσμα να είναι αρκετές οι μέθοδοι και οι αλγόριθμοι που έχουν αναπτυχθεί. Ενώ υπάρχουν ορισμένες μέθοδοι, οι οποίες εστιάζουν στην διερεύνηση των κλάσεων του εκάστοτε προβλήματος ταξινόμησης, στην παρούσα εργασία ασχοληθήκαμε μόνο με αλγορίθμους που εξηγούν τις αποφάσεις του υλοποιημένου CNN για ένα συγκεκριμένο δείγμα/εικόνα [21, 22, 24]. Έπειτα από εκτενή έρευνα στη διαθέσιμη βιβλιογραφία, βρέθηκαν δύο εργασίες που εφαρμόζουν μεθόδους οπτικοποίησης σε προβλήματα ταξινόμησης AD με όμοιο τρόπο. Η πρώτη από αυτές είναι των Korolev et al. [14], στην οποία γίνεται χρήση του αλγορίθμου έγκλεισης σε ένα βαθύ CNN. Παρότι τα αποτελέσματα που λαμβάνουν παρουσιάζουν αρκετές ομοιότητες με τα δικά μας (συγκεκριμένα στις περιοχές που εστιάζει το δίκτυο για να κάνει τις προβλέψεις), δεν γίνεται σύγκριση μεταξύ διαφορετικών μεθόδων ή ανάλυση των χαρτών σχετικότητας με λεπτομέρεια. Η δεύτερη σχετική εργασία είναι των Yang et al [23], οι οποίοι κάνουν χρήση μίας παραλλαγής του αλγορίθμου έγκλεισης που βασίζεται στην κατάτμηση (segmentation) των υποεξέταση περιοχών. Ενώ η συγκεκριμένη μέθοδος παρουσιάζει αρκετές συσχετίσεις με τον αλγόριθμο περιοχικής έγκλεισης που χρησιμοποίησαμε, στην εργασία τους δεν καταφέρνουν να φτάσουν σε ξεκάθαρα συμπεράσματα.

## 2 Δεδομένα

Τα δεδομένα που χρησιμοποιήθηκαν σε αυτή την εργασία αποκτήθηκαν από την διεθνή βάση δεδομένων της ADNI (Alzheimer's Disease Neuroimaging Initiative), κατόπιν σχετικής αίτησης ([adni.loni.usc.edu](http://adni.loni.usc.edu)). Η ADNI ζεκίνησε να δραστηριοποιείται το 2003 ως ένας δημόσιος συνεταιρισμός, υπό την επίβλεψη του Michael W. Weiner. Ο κύριος στόχος αυτής της πρωτοβουλίας μέχρι και σήμερα είναι να διερευνήσει αν η μαγνητική τομογραφία (MRI), η τομογραφία εκπομπής ποζιτρονίων (PET), το λειτουργικό MRI, αλλά και άλλες απεικονιστικές μέθοδοι, σε συνδυασμό με την κλινική διάγνωση, μπορούν να αποτελέσουν εργαλεία που θα μπορούν να διαγνώσουν εγκαίρως την νόσο του Αλτσχάιμερ, με στόχο την καταπολέμησή της.

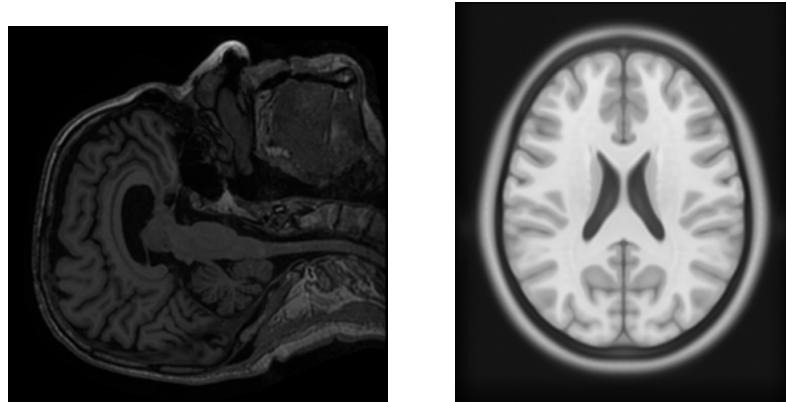
Στην παρούσα εργασία χρησιμοποιήσαμε δεδομένα δομικού MRI (sMRI) από ασθενείς με Αλτσχάιμερ και υγιή άτομα, τα οποία προέρχονται από την πρώτη φάση της ADNI και περιλαμβάνονται στην λίστα “MRI collection - Standardized 1.5T List - Annual 2 year”. Για κάθε ένα υποκείμενο, η λίστα προσφέρει σαρώσεις MRI ολόκληρου του εγκεφάλου και σε τρεις διαφορετικές χρονικές στιγμές (πρώτη εξέταση, 12 μήνες και 24 μήνες μετά την πρώτη εξέταση). Σε ορισμένες περιπτώσεις, μάλιστα, προσφέρονται πολλαπλές σα-

ρώσεις που ανήκουν στην ίδια εξέταση. Από τη δοσμένη λίστα, εξαιρέθηκαν σαρώσεις από άτομα που έπασχαν από ΗΓΔ, καθώς και ορισμένες στις οποίες η προεπεξεργασία τους απέτυχε.

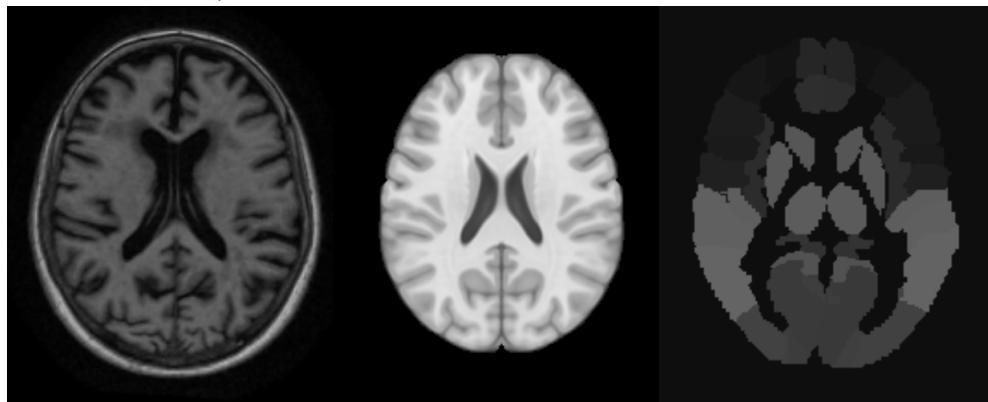
Τα δεδομένα είναι τρισδιάστατες εικόνες T1-weighted MRI. Σε μία εικόνα MRI φαίνονται σε κλίμακα του γκρι οι δομές του εγκεφάλου, ενώ χρησιμοποιούνται δύο ειδών ακολουθίες απεικόνισης, οι T1-weighted και οι T2-weighted ακολουθίες, όπου οι χρόνοι T1, T2 αναπαριστούν τον τρόπο με τον οποίο αντιδρούν οι εγκεφαλικοί ιστοί στο επιβαλλόμενο μαγνητικό πεδίο. Ουσιαστικά επιβάλλεται από τον μαγνητικό τομογράφο ένα μαγνητικό πεδίο, 3 ή 1.5 Tesla (T), προσανατολίζοντας τα μαγνητικά δίπολα των κυττάρων των ιστών. Όταν το μαγνητικό πεδίο σταματήσει να εφαρμόζεται τα δίπολα επιστρέφουν στις αρχικές τους κατευθύνσεις παράγοντας ένα σήμα το οποίο μετράται από ειδικά μηχανήματα, του οποίου οι χρόνοι απόκρισης (T1 ή T2) χρησιμοποιούνται για να συνθέσουν την τελική εικόνα. Μια T1-weighted εικόνα είναι αυτή που η αντίθεσή της εξαρτάται, κυρίως, από τις διαφορές στους χρόνους T1 μεταξύ λίπους και νερού. Στις εικόνες T2-weighted τα υγρά έχουν την μεγαλύτερη ευαισθησία και εμφανίζονται άσπρα, ενώ το νερό και το λίπος που βρίσκονται μέσα στους ιστούς είναι γκρι. Οι εικόνες T1-weighted, συνήθως έχουν πολύ υψηλή ευκρίνεια. Σε αυτές τις εικόνες τα υγρά είναι μαύρα, το νερό που βρίσκεται μέσα στους ιστούς είναι γκρι και οι λιπώδεις ιστοί είναι άσπροι. Παράλληλα, όσο πιο ισχυρό είναι το μαγνητικό πεδίο που χρησιμοποιείται τόσο πιο καλή ανάλυση θα έχει η τελική εικόνα. Στην παρούσα εργασία έγινε χρήση τόσο εικόνων που έχουν προέλθει από σαρώσεις των 3T (180 εικόνες), όσο και εικόνων των 1.5T (969 εικόνες). Στο Σχήμα 2 βλέπουμε παραδείγματα τομών του εγκεφάλου από τα διάφορα στάδια της προεπεξεργασίας των δεδομένων. Είναι άξιο αναφοράς πως μαζί με τις εικόνες MRI, παρέχονταν και κατάλληλα αρχεία κειμένου τα οποία προσφέρουν επιπλέον πληροφορίες για την κάθε μία εικόνα, όπως η ετικέτα (ασθενής ή υγιής), η ημερομηνία επίσκεψης, ο τύπος ακολουθίας της σάρωσης και άλλα, όπως φαίνεται στον Πίνακα 2.

### 3 Υλοποίηση

Το υλοποιημένο σύστημα αποτελείται από τα ακόλουθα μέρη: α) την προεπεξεργασία των δεδομένων σε παραπάνω από ένα στάδια, β) την εκπαίδευση του νευρωνικού δικτύου και γ) την αξιολόγηση της απόδοσής του. Στόχος μας, όπως έχει ήδη αναφερθεί, είναι να καταφέρουμε να ταξινομήσουμε εικόνες μαγνητικής τομογραφίας εγκεφάλων σε δύο κατηγορίες, υγιείς και ασθενείς, με όσο το δυνατόν μεγαλύτερη ακρίβεια. Για το κάθε μέρος έγινε χρήση διαφορετικών εργαλείων, τα οποία θα αναφέρονται κάθε φορά στην αντίστοιχη



(α) Εικόνα MRI, όπως ανακτή- (β) Πρότυπο εικόνας MRI εγκεθηκε από την ADNI, χωρίς επι- φάλου. πλέον επεξεργασία.



(γ) Ευθυγραμμισμένη ει- (δ) Ευθυγραμμισμένη (ε) Άτλαντας αυτόματης κόνα στο πρότυπο εγκε- εικόνα μαλακού ιστού, ανατομικής επισήμανσης φάλου, με χρήση του ερ- με χρήση του εργαλείου (AAL). γαλείου ANTs. FSL.

Σχήμα 2: Δείγματα εικόνων MRI από τα διάφορα στάδια προεπεξεργίας των δεδομένων.

ενότητα.

### 3.1 Επεξεργασία Δεδομένων

Είναι φανερό πως, δεδομένου ότι η εκπαίδευση ενός νευρωνικού δικτύου εξαρτάται άμεσα από τα χαρακτηριστικά των δεδομένων εισόδου, θέλουμε οι είσοδοι μας να είναι όσο το δυνατόν πιο ομοιόμορφοι και χωρίς θόρυβο. Με αυτόν τον τρόπο, το δίκτυο θα βρεί ουσιαστικές συσχετίσεις μεταξύ των διά-

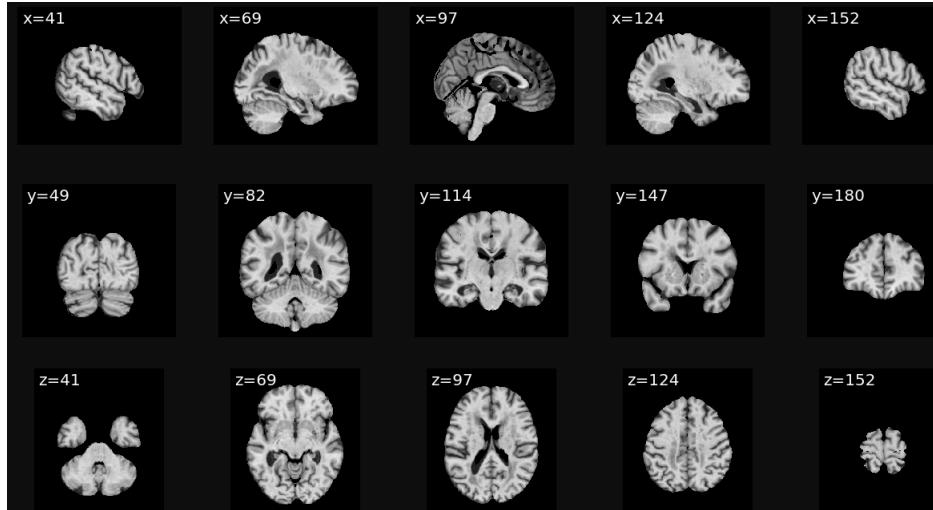
Πίνακας 2: Πίνακες πρόσθετων πληροφοριών εξέτασης από την βάση δεδομένων της ADNI. Η στήλη "PTID" αντιστοιχεί στο αναγνωριστικό του ατόμου, η "VISCODE" στον τύπο της εξέτασης, η "DX" στην ιατρική του κατάσταση, η "EXAMDATE" στην ημερομηνία εξέτασης, η "Image.ID" στο αναγνωριστικό της εικόνας, η "Scan.Date" στην ημερομηνία της σάρωσης, η "Sequence" στον τύπο ακολουθίας της σάρωσης, η "Visit" στην φάση παρακολούθησης, η "Month\_bl" στο χρονικό διάστημα που μεσολαβεί από την πρώτη μέχρι την τωρινή εξέταση και τέλος η "Screen.Diagnosis" στην ιατρική κατάσταση του υποκειμένου μετά τη σάρωση.

PTID	VISCODE	DX	EXAMDATE	Image.ID	Scan.Date	Sequence	Visit	Month_bl	Screen.Diagnosis
011_S_0003	bl	Dementia	2005-09-12	32237	9/1/05	MPR-R; GradWarp; E	Screening	0	AD
011_S_0003	m12	Dementia	2006-09-12	35576	9/12/06	MPR; GradWarp; B1	Month 12	11.9672	AD
011_S_0003	m24	Dementia	2007-09-12	68252	9/12/07	MPR; GradWarp; B1	Month 24	23.9344	AD
011_S_0005	bl	CN	2005-09-07	32246	9/2/05	MPR-R; GradWarp; E	Screening	0	NL
011_S_0005	m12	CN	2006-09-05	31906	9/5/06	MPR; GradWarp; B1	Month 12	11.9016	NL
011_S_0005	m24	CN	2007-09-07	200385	9/7/07	MPR-R; GradWarp; E	Month 24	23.9344	NL
011_S_0010	bl	Dementia	2005-11-10	32270	11/7/05	MPR; GradWarp; B1	Screening	0	AD
011_S_0010	m12	Dementia	2006-11-09	94368	11/9/06	MPR; GradWarp; B1	Month 12	11.9344	AD
011_S_0010	m24	Dementia	2007-11-07	94377	11/7/07	MPR; GradWarp; B1	Month 24	23.8361	AD
022_S_0014	bl	CN	2005-11-04	59375	9/29/05	MPR; GradWarp; B1	Screening	0	NL
022_S_0014	m12	CN	2006-11-27	59391	11/28/06	MPR-R; GradWarp; E	Month 12	12.7213	NL
022_S_0014	m24	CN	2007-11-27	67012	12/6/07	MPR; GradWarp; B1	Month 24	24.6885	NL
011_S_0016	bl	CN	2005-10-13	32306	9/27/05	MPR; GradWarp; B1	Screening	0	NL
011_S_0016	m12	CN	2006-10-11	31933	10/10/06	MPR; GradWarp; B1	Month 12	11.9016	NL

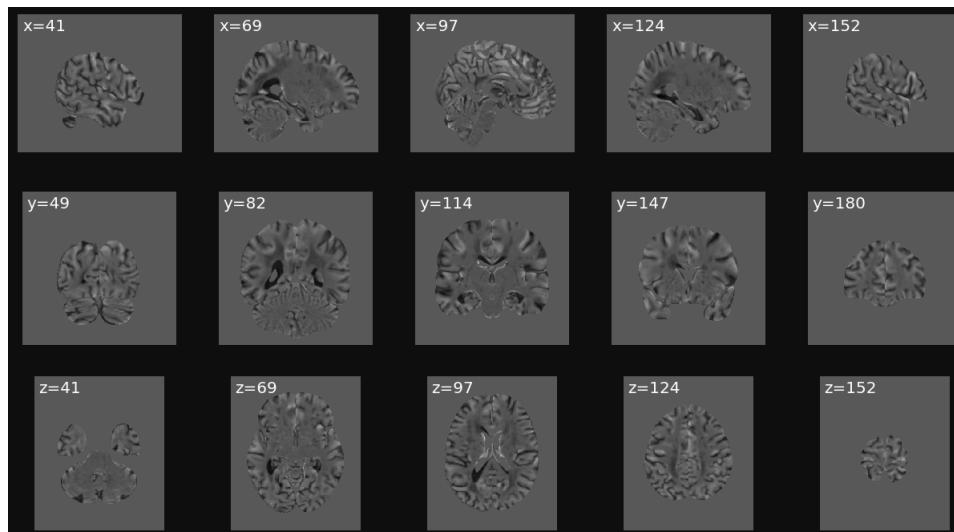
φορων κατηγοριών του προβλήματος πιο γρήγορα και θα καταλήξει σε πιο ορθά συμπεράσματα. Αυτό σημαίνει πως η εκπαίδευσή του θα διαρκέσει λιγότερο και τα ποσοστά ακριβείας θα είναι υψηλότερα. Για να επιτύχουμε το παραπάνω εφαρμόζουμε προεπεξεργασία των δεδομένων. Για το συγκεκριμένο πρόβλημα, εφόσον τα δεδομένα μας είναι εικόνες, θέλουμε να μεγιστοποιήσουμε την χρήσιμη πληροφορία που υπάρχει σε κάθε είσοδο του δικτύου, απομονώνοντας τις περιοχές που μας ενδιαφέρουν, ενώ ταυτόχρονα φροντίζουμε οι εικόνες μεταξύ τους να βρίσκονται σε έναν κοινό άξονα αναφοράς.

Όλες οι εικόνες που προμηθευτήκαμε από την ADNI έχουν υποστεί επεξεργασία, η οποία εστιάζει στην ελαχιστοποίηση του θορύβου και στη διόρθωση της αντίθεσης, όπου αυτό απαιτείται. Ως επόμενο βήμα προεπεξεργασίας, εφαρμόσαμε γραμμική ευθυγράμμιση (linear registration) όλων των δεδομένων σε ένα πρότυπο (template), το οποίο προμηθευτήκαμε από το Κέντρο Απεικόνισης του Εγκεφάλου McConnell, με τη χρήση του εργαλείου ANTs (Advanced Normalization Tools) ([stnava.github.io/ANTs/](https://stnava.github.io/ANTs/)). Σε συνέχεια του προηγούμενου βήματος, εφαρμόσαμε έναν ειδικό αλγόριθμο απομόνωσης του μαλακού ιστού, απομακρύνοντας το κρανίο, με τη χρήση της βιβλιοθήκης εργαλείων ανάλυσης FSL ([fsl.fmrib.ox.ac.uk/](https://fsl.fmrib.ox.ac.uk/)). Οι τελικές εικόνες έχουν την μορφή του Σχήματος 3 και ανάλυση 193x229x193. Σημειώνεται πως η οπτικοποίηση των δεδομένων έγινε με τη χρήση του εργαλείου ιατρικής απεικόνισης Aliza ([www.aliza-dicom-viewer.com/](http://www.aliza-dicom-viewer.com/)). Τέλος, ακολουθεί η κανονικοποίηση

(normalization) των εικόνων σε κοινή κλίμακα φωτεινοτήτων με μέση τιμή 0 και τυπική απόκλιση 1, όπως φαίνεται στο Σχήμα 4.



Σχήμα 3: Ευθυγραμμισμένη εικόνα 3D T1-weighted MRI απομονωμένου μαλακού ιστού. Μία σάρωση αποτελείται από 229 δισδιάστατες εικόνες (άξονες x,y) καλύπτοντας ολόκληρο τον εγκέφαλο (άξονας z). Οι τομές εμφανίζονται σε διάφορες οπτικές γωνίες για λόγους καλύτερης οπτικοποίησης των δεδομένων.



Σχήμα 4: Κανονικοποιημένη και ευθυγραμμισμένη εικόνα MRI μαλακού ιστού.

## 3.2 Μοντέλο

Η αρχιτεκτονική του μοντέλου που υλοποιήθηκε λαμβάνει εμπνεύσεις από το αντίστοιχο των Korolev et al. [14]. Το μοντέλο μας αποτελείται από τέσσερα συνελικτικά επίπεδα (layers), με μέγεθος φίλτρου  $3 \times 3 \times 3$  και χάρτες χαρακτηριστικών (feature maps) πλήθους 8, 16, 32 και 64 αντίστοιχα. Σημειώνεται πως ως χάρτη χαρακτηριστικών ονομάζουμε, ουσιαστικά, το αποτέλεσμα της εφαρμογής των φίλτρων με την είσοδο κάθε επιπέδου. Τα πλήρως συνδεδεμένα επίπεδα (fully connected layers) είναι δύο, με 128 και 64 νευρώνες αντίστοιχα. Θυμίζουμε πως ένα πλήρως συνδεδεμένο επίπεδο αναλαμβάνει να "μεταφράσει" τα χαρακτηριστικά που έχει εξάγει το δίκτυο σε μορφή feature maps, σε παραμέτρους μεμονωμένων νευρώνων, χαρακτηριστικό των κλασσικών βαθιών νευρωνικών δικτύων. Αμέσως μετά από κάθε συνελικτικό επίπεδο, εφαρμόζουμε κανονικοποίηση συνόλου (batch normalization), καθώς και συγκέντρωση (pooling). Η κανονικοποίηση συνόλου κανονικοποιεί τις τιμές των χαρακτηριστικών που έχουν εξαχθεί από το προηγούμενο επίπεδο μετά την συνάρτηση ενεργοποίησης, κάνοντας χρήση της μέσης τιμής των τιμών τους και της τυπικής τους απόκλισης και χρησιμοποιείται για μείωση του χρόνου εκπαίδευσης. Η τεχνική του pooling χρησιμοποιείται για να μειώσει τα χαρακτηριστικά που εξάγονται σε κάθε επίπεδο και κατα συνέπεια την διαστατικότητα του προβλήματος. Παράλληλα, πριν από το πρώτο πλήρως συνδεδεμένο επίπεδο χρησιμοποιείται περιορισμός ενεργοποίησης (dropout) ποσοστού 80%. Αναφορικά με την τεχνική dropout, κατα τη διάρκεια της εκπαίδευσης κάποιες από τις εξόδους του προηγούμενου επιπέδου επιλέγονται τυχαία και αγνοούνται (dropped out). Αυτό έχει ως αποτέλεσμα το συγκεκριμένο επίπεδο σε κάθε επανάληψη της μάθησης να φαίνεται σαν ένα νέο επίπεδο διαφορετικό από τις προηγούμενες επαναλήψεις. Διαισθητικά το δίκτυο μαθαίνει να κοιτάει τα δεδομένα από διαφορετικές οπτικές γωνίες, οι οποίες μεταβάλλονται τυχαία σε κάθε επανάληψη, κάνοντας το μοντέλο πιο ανθεκτικό (robust). Ως συνάρτηση ενεργοποίησης για κάθε επίπεδο χρησιμοποιείται η ReLU (Rectified Linear Unit). Το νευρωνικό δίκτυο έχει δύο νευρώνες εξόδου, των οποίων οι τιμές αντιστοιχούν σε πιθανότητες σχετικές με τις δύο κατηγορίες του προβλήματος (υγιείς - ασθενείς). Την μετατροπή των τιμών των νευρώνων του προτελευταίου επιπέδου στις πιθανότητες του τελευταίου layer αναλαμβάνει η συνάρτηση ενεργοποίησης τύπου softmax. Η συνάρτηση απώλειας που χρησιμοποιείται είναι η cross-entropy, η οποία είναι υπεύθυνη να υπολογίζει τη διαφορά μεταξύ της πρόβλεψης του δικτύου και της πραγματικής τιμής στο τέλος της κάθε επανάληψης. Αντί της κλασσικής διαδικασίας ανανέωσης των βαρών του δικτύου με τη μέθοδο της καθόδου βασισμένης στην κλίση (gradient descent), κάνουμε χρήση του αλγορίθμου βελτιστοποίησης Adam, ο οποίος προτάθηκε το 2014 και συνδυάζει τα πλεονεκτήματα

των αλγορίθμων Adagrad, RMSprop και Stochastic Gradient Descent [11]. Επιπρόσθετα, το κάθε σετ δεδομένων χρησιμοποιείται για να εκπαιδεύσει το δίκτυο σε 20 εποχές (epochs). Σε κάθε εποχή το νευρωνικό δίκτυο “βλέπει” το σετ δεδομένων μία φορά. Το μέγεθος των batches που χρησιμοποιούνται είναι 5, δηλαδή κάθε επανάληψη χαρακτηρίζεται από 5 δείγματα (samples) των συνολικών δεδομένων.

Πίνακας 3: Κάθε σετ δεδομένων χωρίστηκε σε δύο υποσύνολα, ένα για την εκπαίδευση και ένα για την αξιολόγησή του. Παρακάτω φαίνεται το πλήθος των εικόνων του κάθε συνόλου ανά κατηγορία. Σημειώνεται πως η 1η στήλη (Images) του κάθε πίνακα δείχνει τον συνολικό αριθμό εικόνων, ενώ η 4η στήλη (Patients) τον αριθμό των υποκειμένων στο αντίστοιχο σύνολο εικόνων. Οι στήλες 2 και 3 δείχνουν τον αριθμό των εικόνων που αντιστοιχούν σε ασθενείς (AD) ή υγιείς (CN) στο αντίστοιχο σύνολο εικόνων, ενώ οι στήλες 5 και 6 τον αριθμό υποκειμένων της αντίστοιχης κατηγορίας.

	Images	-> AD	-> CN	Patients	-> AD	-> CN
All	180	88	92	65	37	28
Train	73	48	25	25	17	8
Val	107	40	67	40	20	20

(α) Στατιστικά δεδομένων 3T.

	Images	-> AD	-> CN	Patients	-> AD	-> CN
All	969	475	494	344	193	151
Train	794	399	395	284	163	121
Val	175	76	99	60	30	30

(β) Στατιστικά δεδομένων 1.5T.

Τέλος, για την διαδικασία της επαλήθευσης της ακρίβειας του μοντέλου χρησιμοποιούμε την τεχνική της διασταυρωμένης επαλήθευσης k-τμημάτων (k-fold cross validation) με  $k = 5$ . Κατά τη διάρκεια του cross validation, ολόκληρο το σετ δεδομένων χωρίζεται σε k τμήματα και έπειτα, επαναληπτικά, χρησιμοποιούμε το κάθε τμήμα των δεδομένων για την αξιολόγησή του, ενώ τα υπόλοιπα τμήματα χρησιμοποιούνται στην εκπαίδευση. Ο μέσος όρος όλων των επαναλήψεων της παραπάνω διαδικασίας δίνει την μετρική ακρίβειας του μοντέλου. Τα στατιστικά των δεδομένων φαίνονται στον Πίνακα 3i για τις σαρώσεις των 3T και στον Πίνακα 3ii για τις αντίστοιχες των 1.5T. Το μοντέλο που υλοποιήσαμε για τους σκοπούς της εργασίας, δημιουργήθηκε στο framework PyTorch ([pytorch.org/](http://pytorch.org/)).

### 3.3 Οπτικοποίηση

Σε αυτή την ενότητα γίνεται μία σύντομη περιγραφή των τεσσάρων μεθόδων οπτικοποίησης που χρησιμοποιήθηκαν σε αυτή την εργασία. Για μία πιο αναλυτική περιγραφή τεχνικών και μεθόδων οπτικοποίησης μπορεί κανείς να απευθυνθεί στο άρθρο ανασκόπησης των Montavon et al. [27]. Κάθε ένας από τους ακόλουθους αλγόριθμους παράγει ως έξοδο έναν χάρτη θερμοτήτων (heatmap) πάνω στην εικόνα εισόδου, ο οποίος επισημαίνει την σχετικότητα του κάθε εικονοστοιχείου (pixel) με την τελική απόφαση της ταξινόμησης.

1) *Sensitivity Analysis (Backpropagation)* [21]: Σε αυτή τη μέθοδο υπολογίζεται η κλίση (gradient) της πιθανότητας εξόδου του δικτύου σε συνάρτηση με την εικόνα εισόδου. Για κάθε εικονοστοιχείο της εικόνας, αυτή η κλίση περιγράφει πόσο μεταβάλλεται η πιθανότητα εξόδου όταν αλλάζει η τιμή του pixel. Στα νευρωνικά δίκτυα, το gradient μπορεί να υπολογιστεί μέσω του αλγορίθμου οπισθοδιάδοσης (backpropagation), ο οποίος χρησιμοποιείται για την εκαπίδευση του δικτύου. Ως μετρική σχετικότητας, παίρνουμε την απόλυτη τιμή του gradient.

2) *Guided Backpropagation* [22]: Αυτή η τεχνική αποτελεί μία τροποποιημένη έκδοση της μεθόδου Sensitivity Analysis, στην οποία τα αρνητικά gradients μηδενίζονται στα επίπεδα ενεργοποίησης (ReLU layers) κατά τη διάρκεια του οπίσθιου περάσματος (backward pass). Ουσιαστικά αποτελεί έναν συνδυασμό των μεθόδων οπισθοδιάδοσης και αποσυνέλιξης. Βασικό της πλεονέκτημα είναι οι καλύτερα “εστιασμένοι” χάρτες θερμοτήτων. Όπως και στην προηγούμενη μέθοδο, χρατάμε την απόλυτη τιμή του gradient.

3) *Occlusion* [24]: Σε αυτή τη μέθοδο ένα μέρος της εικόνας αποκλείεται μέσω ενός μαύρου ή γκρι πλαισίου (patch) και η έξοδος του δικτύου επανυπολογίζεται. Αν η πιθανότητα για την σωστή κλάση ταξινόμησης μειώθει, συγχριτικά με την πιθανότητα που προέκυψε από την αρχική εικόνα, τότε αυτή η περιοχή της εικόνας θεωρείται σχετική με την τελική απόφαση. Για να προκύψει το τελικό heatmap, το πλαίσιο μετακινείται κατά μήκος ολόκληρης της εικόνας και σχεδιάζεται η διαφορά μεταξύ των πιθανοτήτων που προέκυψαν πριν και μετά την έγκλειση. Το πλαίσιο που χρησιμοποιούμε έχει μέγεθος  $40 \times 40 \times 40$  με τιμή 0.

4) *Brain Area Occlusion*: Αυτός ο αλγόριθμος αποτελεί μία τροποποιημένη έκδοση της τεχνικής Occlusion, στον οποίο αποκλείουμε μία ολόκληρη περιοχή του εγκεφάλου με βάση τον άτλαντα αυτόματης ανατομικής επισήμανσης (Automated Anatomical Labeling atlas - AAL atlas) ([www.gin.cnrs.fr/en](http://www.gin.cnrs.fr/en)),

όπως φαίνεται στο Σχήμα 2ε. Αυτή η μέθοδος είναι εμπνευσμένη από την τεχνική οπτικοποίησης των Yang et al. [23], που είναι βασισμένη στην τμηματοποίηση (segmentation). Στον τελικό heatmap περιλαμβάνεται η διαφορά των πιθανοτήτων πριν και μετά την έγκλειση, όπως και στην προηγούμενη τεχνική.

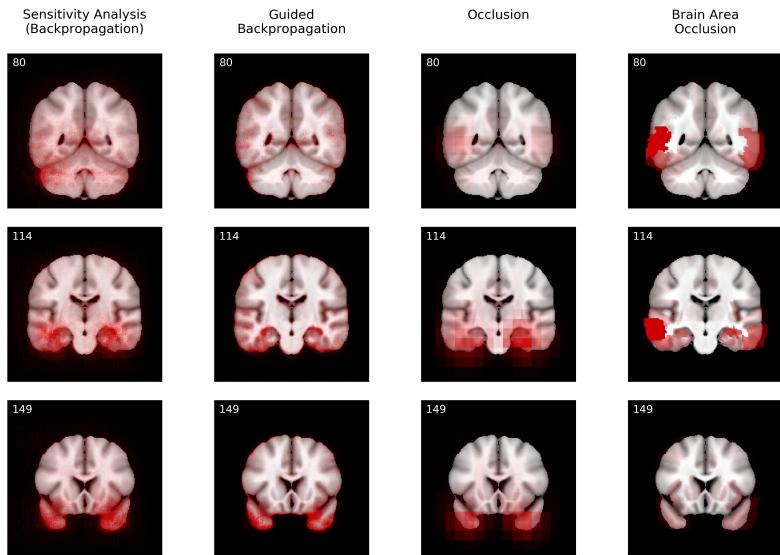
## 4 Αποτελέσματα

Στόχος της εργασίας ήταν αφενός η δημιουργία ενός συστήματος διάγνωσης της νόσου Αλτσχάιμερ με την χρήση εικόνων μαγνητικής τομογραφίας, αφετέρου η οπτικοποίηση των αποφάσεων του δικτύου, η ερμηνεία αυτών και η σύγκρισή τους με την διαθέσιμη βιβλιογραφία.

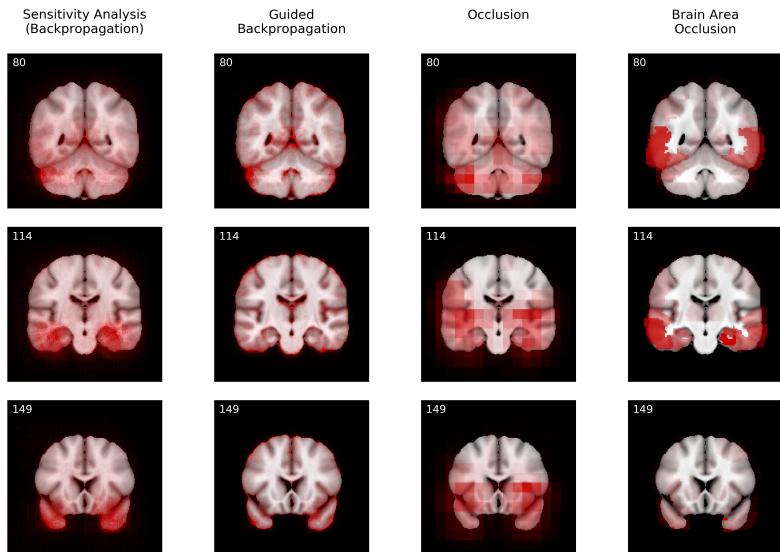
### 4.1 Ταξινόμηση

Κάνοντας χρήση της τεχνικής 5-fold cross validation, όπως αυτή περιγράφηκε στην προηγούμενη ενότητα, μπορέσαμε να αξιολογήσουμε την επίδοση του νευρωνικού δικτύου στο δεδομένο πρόβλημα ταξινόμησης. Αναφορικά με το εκπαιδευμένο δίκτυο στο σετ δεδομένων των 1.5 Tesla, το δίκτυο επιτυγχάνει ακρίβεια ταξινόμησης  $0.77 \pm 0.06$ . Το αποτέλεσμα αυτό είναι συγχρίσιμο με τα αντίστοιχα αποτελέσματα που προέκυψαν από τα μοντέλα συνελικτικών νευρωνικών δικτύων άλλων πρόσφατων ερευνών [8, 26, 14]. Για παράδειγμα, οι Korolev et al. [14], οι οποίοι χρησιμοποιούν παρόμοιο μοντέλο και διαδικασία εκπαίδευσης, επιτυγχάνουν παρόμοια ποσοστά ακρίβειας  $0.79 \pm 0.08$ . Παράλληλα, το δίκτυο που εκπαιδεύτηκε στα δεδομένα των 3 Tesla, πέτυχε ακρίβεια ταξινόμησης  $0.69 \pm 0.05$ . Γίνεται εμφανές από τα παραπάνω, πως η καλύτερη ανάλυση των εικόνων που έχουν προέλθει από σαρώσεις μεγαλύτερου μαγνητικού πεδίου, δεν είναι αρκετή για να αναπληρώσει το μικρότερο πλήθος δειγμάτων του συγκεκριμένου σετ. Να σημειωθεί επίσης ότι οι εικόνες 3T φαίνεται να περιέχουν μεγαλύτερο ποσοστό θορύβου, αφού τα υποκείμενα μένουν για μικρότερο χρονικό διάστημα μέσα στον σαρωτή.

Σημειώνεται πως πολλές έρευνες έχουν ως κύριο στόχο την βελτιστοποίηση της ακρίβειας ταξινόμησης, με αποτέλεσμα να κάνουν χρήση πιο σύνθετων μοντέλων, σημαντικά μεγαλύτερων σετ από δεδομένα και περισσότερων από ένα μηχανημάτων απεικόνισης (MRI, PET, DTI, fMRI). Η παρούσα εργασία ωστόσο επικεντρώθηκε στην δημιουργία ενός συστήματος, του οποίου τα αποτελέσματα είναι εύκολο να ερμηνευτούν και η αναπαραγωγή τους είναι γρήγορη και χωρίς να απαιτεί μεγάλο πλήθος υπολογιστικών πόρων. Έτσι, καθίσταται πιο εύκολη η επέκτασή της από μελλοντικούς ερευνητές.



(α) Χάρτες σχετικότητας εγκεφαλικών περιοχών για τα δεδομένα των ασθενών.



(β) Χάρτες σχετικότητας εγκεφαλικών περιοχών για τα δεδομένα των υγιών υποκειμένων.

**Σχήμα 5:** Χάρτες σχετικότητας για όλες τις μεθόδους οπτικοποίησης, υπολογισμένοι κατά μέσο όρο για όλα τα δείγματα εικόνων των ασθενών (i) και των υγιών υποκειμένων (ii), όπως αυτά έχουν ληφθεί από την ADNI. Το χόκκινο χρώμα δείχνει το βαθύτερο σχετικότητας, δηλαδή το πόσο η δεδομένη (εγκεφαλική) περιοχή ήταν σημαντική στην τελική απόφαση ταξινόμησης του δικτύου. Οι αριθμοί δηλώνουν τις θέσεις των εν λόγω τομών.

## 4.2 Σχετικές εγκεφαλικές περιοχές

Στα Σχήματα 5α και 5β φαίνονται οι χάρτες σχετικότητας για όλες τις μεθόδους οπτικοποίησης, υπολογισμένοι κατά μέσο όρο για όλα τα δείγματα εικόνων των ασθενών και των υγιών αντίστοιχα. Δεδομένου ότι δεν υπάρχει διαθέσιμη μέθοδος επαλήθευσης για τους εν λόγω χάρτες, η αξιολόγηση των αποτελεσμάτων θα γίνει εστιάζοντας σε συγκεκριμένες περιοχές του εγκεφάλου που είναι γνωστό από την βιβλιογραφία ότι συνδέονται με τη νόσο του Αλτσχάιμερ. Ξεχωρίσαμε τις πιο σχετικές εγκεφαλικές περιοχές για κάθε αλγόριθμο οπτικοποίησης αθροίζοντας την σχετικότητα που προέκυψε από τους χάρτες σε κάθε περιοχή, με βάση τον άτλαντα AAL. Στον Πίνακα 4 φαίνονται οι τέσσερις πιο σχετικές περιοχές για κάθε μέθοδο, υπολογισμένες κατά μέσο όρο για όλα τα δείγματα εικόνων των δύο κατηγοριών ταξινόμησης.

Τόσο για τα δεδομένα των ασθενών όσο και για τα δεδομένα των υγιών ατόμων, μπορούμε να δούμε πως η βασική περιοχή εστίασης του δικτύου είναι αυτή του χροταφικού λοβού, και ιδιαίτερα το μεσαίο τμήμα του. Αυτή η περιοχή, η οποία περιλαμβάνει τον ιππόκαμπο και άλλες δομές σχετικές με τη μνήμη, έχει συνδεθεί εμπειρικά με τη νόσο του Αλτσχάιμερ [28]. Η περιοχή του ιπποκάμπου είναι συνήθως μία από τις περιοχές του εγκεφάλου που προσβάλλονται το νωρίτερο από τη νόσο [29]. Στη μελέτη μας, ενώ παρατηρούμε υψηλό βαθμό σχετικότητας στην περιοχή του ιπποκάμπου, συνήθως, ολόκληρη η περιοχή γύρω του φαίνεται να είναι κρίσιμης σημασίας για την απόφαση του δικτύου. Αυτό, πιθανόν, μπορεί να εξηγηθεί από το γεγονός ότι τα δεδομένα μας αποτελούνται μόνο από περιπτώσεις στις οποίες η νόσος βρίσκεται σε προχωρημένο στάδιο.

Εκτός από τις χροταφικές περιοχές, παρατηρούμε σχετικότητα και σε άλλες περιοχές κατά μήκος ολόκληρου του εγκεφάλου (ειδικά στις βασισμένες στην κλίση (gradient-based) μεθόδους οπτικοποίησης). Βρίσκουμε πως η κατανομή της σχετικότητας ποικίλει ανάμεσα στους ασθενείς. Σε ορισμένους εγκεφάλους έχουμε ισχυρή σχετικότητα στον χροταφικό λοβό (temporal lobe), ενώ σε άλλους ο φλοιός (cortex) παίζει πολύ σημαντικό ρόλο.

Τέλος, σημειώνουμε πως οι χάρτες σχετικότητας για τους ασθενείς και τους υγιείς εμφανίζουν αρκετές ομοιότητες. Το παραπάνω φαίνεται λογικό, αν σκεφτεί κανείς πως το δίκτυο πρέπει να εστιάζει στις ίδιες εγκεφαλικές περιοχές για να εντοπίσει την ύπαρξη ή απουσία της νόσου. Παράλληλα, φαίνεται να υπάρχουν ορισμένες διαφορές μεταξύ των ασθενών και των υγιών στα αντίστοιχα heatmaps, τα οποία έχουν υπολογιστεί με τη χρήση της μεθόδου έγκλεισης. Αυτό ενδεχομένως να οφείλεται στο πλαίσιο έγκλεισης που χρησιμοποιείται, καθώς είναι αρκετά πιθανό το δίκτυο να το αντιλαμβάνεται ως εγκεφαλική ατροφία αυξάνοντας την σχετικότητα ορισμένων περιοχών.

Πίνακας 4: Οι πιο σχετικές εγκεφαλικές περιοχές για όλα τα δείγματα εικόνων των ασθενών και των υγιών υποκειμένων. Οι τιμές στις παρενθέσεις δηλώνουν το ποσοστό της αθροισμένης σχετικότητας της δεδομένης περιοχής ως προς την αθροισμένη σχετικότητα ολόκληρου του εγκεφάλου.

	Sensitivity Analysis (Backpropagation)	Guided Backpropagation	Occlusion	Brain Area Occlusion
Ασθενείς	Μέσος Κροταφικός (6.1%) Άνω Κροταφικός (5.9%) Ατρακτοειδής (4.6%) Σκέλος Ι Παρεγκεφαλίδας (3.8%)	Μέσος Κροταφικός (7.0%) Άνω Κροταφικός (5.7%) Μέσος Μετωπιαίος (4.2%) Ατρακτοειδής (3.9%)	Μέσος Κροταφικός (12.1%) Άνω Κροταφικός (9.2%) Ατρακτοειδής (6.2%) Παράπονοκάμπια Έλικα (5.4%)	Μέσος Κροταφικός (29.7%) Άνω Κροταφικός (14.8%) Κάτω Κροταφικός (4.4%) Ιππόκαμπος (4.1%)
	Μέσος Κροταφικός (6.1%) Άνω Κροταφικός (5.8%) Ατρακτοειδής (4.5%) Σκέλος Ι Παρεγκεφαλίδας (3.8%)	Σκέλος Ι Παρεγκεφαλίδας (4.6%) Μέσος Κροταφικός (4.5%) Άνω Κροταφικός (4.5%) Μέσος Μετωπιαίος (4.1%)	Μέσος Κροταφικός (6.2%) Κάτω Κροταφικός (4.9%) Σκέλος Ι Παρεγκεφαλίδας (4.9%) Νήσος (4.7%)	Μέσος Κροταφικός (20.4%) Άνω Κροταφικός (12.8%) Ατρακτοειδής (7.2%) Κάτω Κροταφικός (6.2%)

Πίνακας 5: Ευκλείδια απόσταση μεταξύ των χαρτών σχετικότητας κατά μέσο όρο για όλα τα δείγματα ασθενών και υγιών υποκειμένων. Σημειώνεται πως οι παρακάτω τιμές είναι σε κλίμακα  $10^{-4}$ .

	Sensitivity Analysis (Backpropagation)	Guided Backpropagation	Occlusion	Brain Area Occlusion
Sensitivity Analysis (Backpropagation)	0.00 / 0.00	4.09 / 4.36	5.15 / 4.09	11.48 / 9.04
Guided Backpropagation		0.00 / 0.00	6.47 / 5.83	11.36 / 9.80
Occlusion	5.15 / 4.09		6.47 / 5.83	0.00 / 0.00
Brain Area Occlusion	11.48 / 9.04		11.36 / 9.80	11.16 / 9.66
				0.00 / 0.00

### 4.3 Διαφορές μεταξύ των μεθόδων οπτικοποίησης

Ενώ όλες οι μέθοδοι οπτικοποίησης εστιάζουν σε παρόμοιες εγκεφαλικές περιοχές, υπάρχουν ορισμένες διαφορές μεταξύ τους που αξίζει να αναφέρουμε. Οι τεχνικές έγκλεισης και περιοχικής έγκλεισης εστιάζουν περισσότερο σε συγκεκριμένες περιοχές, ενώ η σχετικότητα στις τεχνικές που είναι βασισμένες στην κλίση (gradient) φαίνεται να είναι αρκετά πιο κατανεμημένη στο σύνολο του εγκεφάλου. Οι μέθοδοι που είναι βασισμένες στην κλίση δεν μπορούν να χειρίστούν μεγάλες περιοχές κατανεμημένης σχετικότητας, όπως ο φλοιός, καθώς αυτές οι περιοχές δεν πρόκειται να καλυφθούν ποτέ πλήρως από το πλαίσιο έγκλεισης. Επομένως, προτείνεται η χρήση gradient-based μεθόδων σε περιπτώσεις χρήσης, όπου η σχετικότητα αναμένεται να είναι κατανεμημένη σε ολόκληρη την εικόνα εισόδου. Επιπρόσθετα, βρίσκουμε πως ενώ η περιοχική έγκλειση αποτελεί μία αρκετά φυσική προσέγγιση στο δεδομένο πρόβλημα, χαρακτηρίζεται από ένα πολύ βασικό μειονέκτημα: μπορεί να καλύψει μόνο μία εγκεφαλική περιοχή τη φορά. Στην περίπτωσή μας, αυτό οδηγεί σε πολύ υψηλή σχετικότητα στον κροταφικό λοβό, αλλά ελάχιστη έως καθόλου σε άλλες δομές του εγκεφάλου.

Για να συγχρίνουμε τις μεθόδους οπτικοποίησης ποσοτικά, υπολογίσαμε την Ευκλείδια απόσταση μεταξύ όλων των τελικών χαρτών σχετικότητας ( $\sqrt{\sum_i (A_i - B_i)^2}$ , όπου  $A$  και  $B$  είναι τα heatmaps δύο διαφορετικών με-

θόδων και *i* η τοποθεσία του κάθε ογκοστοιχείου). Οι αποστάσεις φαίνονται στον Πίνακα 5. Σε αντιστοιχία με την ποιοτική σύγκριση των μεθόδων βρίσκουμε πως οι gradient-based τεχνικές είναι αρκετά όμοιες μεταξύ τους, όπως φαίνεται από την μικρή Ευκλείδια απόσταση μεταξύ τους. Η μόνη μέθοδος που φαίνεται να απέχει αρκετά από τις υπόλοιπες μεθόδους, είναι η περιοχική έγκλειση, η οποία όπως αναφέραμε παραπάνω αποδίδει σχετικότητα μόνο σε ορισμένες περιοχές.

## 5 Συμπεράσματα

Σε αυτή την εργασία, εκπαιδεύσαμε ένα τρισδιάτατο συνελικτικό νευρωνικό δίκτυο για την διάγνωση της νόσου του Αλτσχάιμερ μέσω ταξινόμησης και εφαρμόσαμε διάφορες μεθόδους οπτικοποίησης των αποφάσεών του. Δείχναμε πως το δίκτυο εστιάζει σε εγκεφαλικές περιοχές που είναι εμπειρικά συνδεδεμένες με τη νόσο, όπως ο μέσος κροταφικός λοβός, κάτι το οποίο φαίνεται και από τις τέσσερις τεχνικές οπτικοποίησης. Ενδιαφέρον είναι το γεγονός ότι η κατανομή της σχετικότητας ποικίλλει ανάμεσα στους ασθενείς, με ορισμένους να εμφανίζουν μεγαλύτερη σχετικότητα στον κροταφικό λοβό, ενώ άλλοι σε πιο φλοιϊκές περιοχές. Δείχνουμε πως η εφαρμογή αλγορίθμων οπτικοποίησης σε νευρωνικά δίκτυα αποτελεί ένα σημαντικό βήμα στην κατανόηση των αποφάσεών τους, κάτι το οποίο μπορεί να αυξήσει την κλινική εμπιστοσύνη στα υπολογιστικά συστήματα υποστήριξης λήψης αποφάσεων και να οδηγήσει στην εμφάνιση μοντέλων μηχανικής μάθησης στον τομέα της ιατρικής. Τα αποτελέσματά μας δείχνουν ακόμα, πως οι διαφορετικές μέθοδοι οπτικοποίησης διαφέρουν στην ερμηνεία των αποφάσεων ενός νευρωνικού δικτύου. Επομένως, συνίσταται η σύγκριση των διαθέσιμων τεχνικών οπτικοποίησης για τη δεδομένη εφαρμογή, και όχι η επιλογή μιας μεθόδου με βάση τα αποτελέσματά της σε ένα διαφορετικό πλαίσιο.

Διακρίνουμε τρεις πιθανούς άξονες για μελλοντική έρευνα: Πρώτον, την επέκταση σε άλλες μεθόδους οπτικοποίησης [27] και την σύγκρισή τους με τα αποτελέσματα της εργασίας μας. Δεύτερον, τη δημιουργία ενός δικτύου διάγνωσης άλλα και πρόβλεψης της νόσου του Αλτσχάιμερ, με την ενσωμάτωση δεδομένων από ασθενείς με Ήπια Γνωστική Δυσλειτουργία. Τρίτον, θα ήταν ιδιαίτερα ενδιαφέρον να υλοποιηθεί μία μέθοδος επαλήθευσης των χαρτών σχετικότητας υπό τη μορφή μιας αλήθειας βασισμένης σε δεδομένα και όχι ως αποτέλεσμα συμπεράσματος (ground truth), πιθανότατα με την ενσωμάτωση κατάλληλων μοντέλων προσομοίωσης.

## Αναφορές

- [1] Magnin B., Mesrob L., Kinkingnehu S., Pelegrini-Issac M., Colliot O., Sarazin M., Dubois B., Lehericy S., Benali H. *Support vector machine-based classification of alzheimers disease from whole-brain anatomical mri.* Neuroradiology 51, 73–83 (2009).
- [2] Ahmed O.B., Mizotin M., Benois-Pineau J., Allard M., Catheline G., Amar C.B. *Alzheimer's disease diagnosis on structural mr images using circular harmonic functions descriptors on hippocampus and posterior cingulate cortex.* Computerized Medical Imaging and Graphics 44, 13–25 (2015).
- [3] Ebadi A., Dalboni da Rocha J.L., Nagaraju D.B., Tovar-Moll F., Bramati I., Coutinho G., Sitaram R., Rashidi P. *Ensemble classification of alzheimer's disease and mild cognitive impairment based on complex graph measures from diffusion tensor images.* Frontiers in Neuroscience 11 (2017).
- [4] Lee W., Park B., Han K. *Svm-based classification of diffusion tensor imaging data for diagnosing alzheimers disease and mild cognitive impairment.* International Conference on Intelligent Computing 489–499 (2015).
- [5] Lei B., Chen S., Ni D., Wang T. *Discriminative learning for alzheimer's disease diagnosis via canonical correlation analysis and multimodal fusion.* Frontiers in aging neuroscience 8 (2016).
- [6] Ahmed O.B., Benois-Pineau J., Allard M., Amar C.B., Catheline G. *Classification of alzheimers disease subjects from MRI using hippocampal visual features.* Multimedia Tools and Applications 74, 1249–1266 (2015).
- [7] Vu T.D., Yang H.J., Nguyen V.Q., Oh A.R., Kim M.S. *Multimodal learning using convolution neural network and sparse autoencoder.* IEEE International Conference 309–312 (2017).
- [8] Payan A., Montana G. *Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks.* arXiv preprint (2015).
- [9] Glozman T., Liba o. *Hidden cues: Deep learning for Alzheimers disease classification.*

- [10] Sarraf S., Tofighi G. *Classification of Alzheimer's disease structural mri data by deep learning convolutional neural networks*. arXiv preprint (2016).
- [11] Billones C.D., Demetria O.J.L.D., Hostallero D.E.D., Naval P.C. *Demnet: A convolutional neural network for the detection of Alzheimer's disease and mild cognitive impairment*. Region 10 Conference (TENCON) IEEE 3724–3727 (2016).
- [12] Aderghal K., Benois-Pineau J., Afdel K., Gwenaelle C. *Fuseme: Classification of sMRI images by fusion of deep cnns in 2d+eps projections*. International Workshop on Content-Based Multimedia Indexing 1–7 (2017).
- [13] Shi J., Zheng X., Li Y., Zhang Q., Ying S. *Multimodal neuroimaging feature learning with multimodal stacked deep polynomial networks for diagnosis of alzheimer's disease*. IEEE journal of biomedical and health informatics (2017).
- [14] Korolev S., Safullin A., Belyaev M., Dodonova Y. *Residual and plain convolutional neural networks for 3D brain MRI classification*. ISBI (2017).
- [15] Suk H.I., Lee S.W., Shen D. *Deep ensemble learning of sparse regression models for brain disease diagnosis*. Medical image analysis 37, 101–113 (2017).
- [16] Luo S., Li X., Li J. *Automatic alzheimers disease recognition from mri data using deep learning method*. Journal of Applied Mathematics and Physics 5 (2017).
- [17] Wang S., Shen Y., Chen W., Xiao T., Hu J. *Automatic recognition of mild cognitive impairment from mri images using expedited convolutional neural networks*. International Conference on Artificial Neural Networks 373–380 (2017).
- [18] Li H., Habes M., Fan Y. *Deep ordinal ranking for multi-category diagnosis of alzheimer's disease using hippocampal mri data*. arXiv preprint (2017).
- [19] Li X., Li Y., Li X. *Predicting clinical outcomes of alzheimers disease from complex brain networks*. International Conference on Advanced Data Mining and Applications 519–525 (2017).

- [20] Cheng D., Liu M., Fu J., Wang Y. *Classification of mr brain images by combination of multi-cnns for ad diagnosis*. Ninth International Conference on Digital Image Processing (ICDIP) 1042042 (2017).
- [21] Simonyan, K. Vedaldi A., Zisserman A. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. arXiv preprint (2014).
- [22] Springenberg J.T., Dosovitskiy A., Brox T., Riedmiller M. *Striving for Simplicity: The All Convolutional Net*. ICLR (2015).
- [23] Yang C. Rangarajan A. Ranka S. *Visual Explanations From Deep 3D Convolutional Neural Networks for Alzheimer's Disease Classification*. arXiv preprint (2018).
- [24] Zeiler M.D., Fergus R. *Visualizing and understanding convolutional networks*. ECCV (2014).
- [25] Khvostikov A., Aderghal K., Benois-Pineau J., Krylov A., Catheline G. *3D CNN-based classification using smRI and MD-DTI images for Alzheimer disease studies*. arXiv preprint (2018).
- [26] Hosseini-Asl E., Gimelfarb G., El-Baz A. *Alzheimer's Disease Diagnostics by a Deeply Supervised Adaptable 3D Convolutional Network*. Frontiers in Bioscience 23, 584–596 (2018).
- [27] Montavon G., Samek, W. Muller K.R. *Methods for interpreting and understanding deep neural networks*. Digital Signal Processing 73, 1–15 (2018).
- [28] Frisoni G.B., Fox N.C., Jack C.R., Scheltens P., Thompson P.M. *The clinical use of structural MRI in Alzheimer disease*. Nature Reviews Neurology 6, 67–77 (2010).
- [29] Mu Y., Gage F.H. *Adult hippocampal neurogenesis and its role in Alzheimer's disease*. Molecular Neurodegeneration 6, 85 (2011).

## A Παράρτημα

Παράκατω βρίσκονται τα δύο jupyter notebooks τα οποία υλοποιούν την εκπαίδευση του δικτύου και την οπτικοποίησή του. Οι συναρτήσεις που χρησιμοποιούνται σε αυτά, καθώς και ο πηγαίος κώδικας ολόκληρης της εργασίας μαζί με το κατάλληλο documentation, μπορούν να βρεθούν στον σύνδεσμο [github.com/iliaspan/cnn-adni](https://github.com/iliaspan/cnn-adni).

# cnn\_training

December 10, 2019

## 1 Load data

```
[0]: from google.colab import drive  
drive.mount('/content/gdrive')
```

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call  
drive.mount("/content/gdrive", force\_remount=True).

```
[0]: cd gdrive/My\ Drive/Demokritos-INN
```

/content/gdrive/My Drive/Demokritos-INN

```
[0]: from __future__ import absolute_import, division, print_function, u  
      ↪unicode_literals  
  
%load_ext autoreload  
%autoreload 2  
  
import numpy as np  
import scipy as sp  
import matplotlib.pyplot as plt  
import matplotlib as mpl  
import pandas as pd  
%matplotlib inline  
%config InlineBackend.figure_format = 'retina' # adapt plots for retina  
      ↪displays  
  
from IPython.core.debugger import set_trace
```

```
[0]: import os  
import utils  
from tqdm import tqdm_notebook  
from sklearn.model_selection import train_test_split  
import multiprocessing  
import json
```

```
[0]: import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.autograd import Variable
from torch.utils.data import Dataset, DataLoader

[0]: pip install git+https://github.com/iliasanpan/torchsample

[0]: import datasets, models

[0]: # Load the data table with all 3 Tesla MRI scans from ADNI.
df = datasets.load_data_table_3T()
```

Loading dataframe for data/ADNI/ADNI\_tables/customized/DxByImgClean\_CompleteAnnual2YearVisitList\_3T.csv  
 Found 279 images in table  
 Filtered out 1 of 279 images because of failed preprocessing  
 Filtered out 98 of 278 images that were MCI  
 Final dataframe contains 180 images from 65 patients

```
[0]: df.head()

[0]:      RID   ...           filepath
0    15   ...  data/ADNI/ADNI_2Yr_3T_preprocessed/I33046.nii.gz
1    15   ...  data/ADNI/ADNI_2Yr_3T_preprocessed/I63475.nii.gz
2    15   ...  data/ADNI/ADNI_2Yr_3T_preprocessed/I82551.nii.gz
5    30   ...  data/ADNI/ADNI_2Yr_3T_preprocessed/I138580.nii.gz
6    30   ...  data/ADNI/ADNI_2Yr_3T_preprocessed/I87493.nii.gz

[5 rows x 14 columns]

[0]: # Patient-wise train-test-split.
# Select a number of patients for each class, put all their images in the test set
# and all other images in the train set. This is the split that is used in the paper to produce the heatmaps.
test_patients_per_class = 20

patients_AD = df[df['DX'] == 'Dementia']['PTID'].unique()
patients_CN = df[df['DX'] == 'CN']['PTID'].unique()
#patients_CN = filter(lambda p: p not in patients_AD, patients_CN) # patients that have both a CN and an AD scan should belong to the AD group

[0]: patients_AD_train, patients_AD_test = train_test_split(patients_AD, test_size=test_patients_per_class, random_state=0)
patients_CN_train, patients_CN_test = train_test_split(patients_CN, test_size=test_patients_per_class, random_state=0)
```

```
[0]: patients_train = np.concatenate([patients_AD_train, patients_CN_train])
patients_test = np.concatenate([patients_AD_test, patients_CN_test])

[0]: train_dataset, test_dataset = datasets.build_datasets(df, patients_train, patients_test)
```

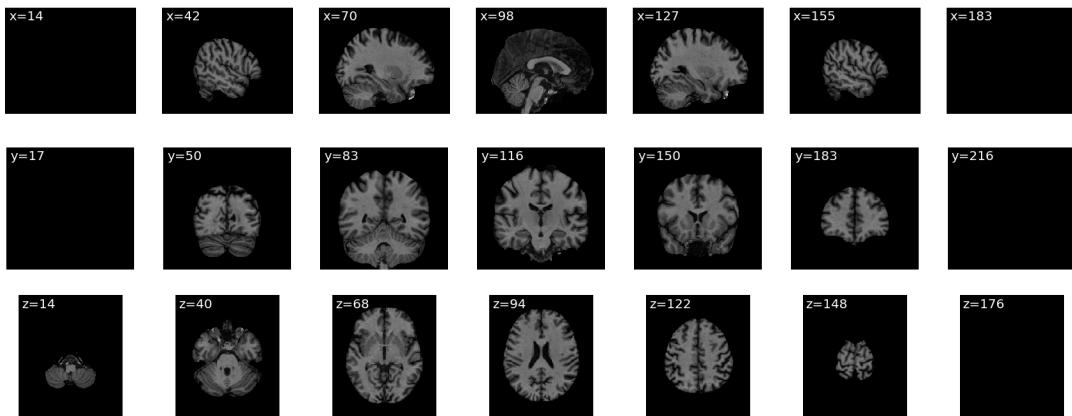
	Images	-> AD	-> CN	Patients	-> AD	-> CN
All	180	88	92	65	37	28
Train	73	48	25	25	17	8
Val	107	40	67	40	20	20

Calculating mean and std for normalization:

```
HBox(children=(IntProgress(value=0, max=60), HTML(value='')))
```

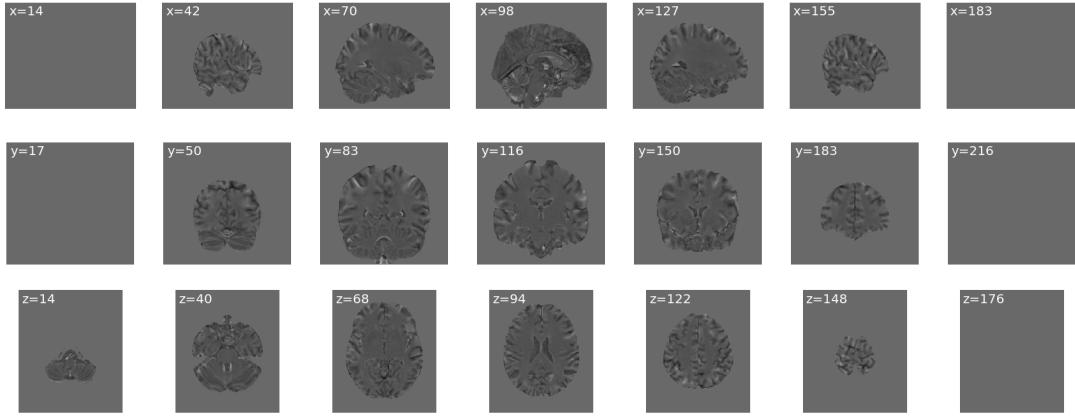
```
[0]: # Plot one MRI scan from the dataset, first without normalization...
i = 0
utils.plot_slices(train_dataset.get_raw_image(i))
```

0.0 629.094449098244 None None



```
[0]: # ... and then with normalization, i.e. the way the network sees it.
utils.plot_slices(train_dataset[i][0][0])
```

tensor(-2.0476) tensor(2.8597) None None



```
[0]: train_loader, test_loader = datasets.build_loaders(train_dataset, test_dataset)
```

## 2 Create model and train

```
[0]: # This creates the model from the paper in pytorch, and wraps it in a `trainer` ↴ via torchsample.  
net, trainer, cuda_device = models.build_model()
```

Moved network to GPU

```
[0]: models.train_model(trainer, train_loader, test_loader, cuda_device, ↴  
    num_epoch=10)
```

```
Epoch 1/10: 0% | 0/19 [00:00<?, ? batches/s]/content/gdrive/My  
Drive/Demokritos-INN/models.py:198: UserWarning: Implicit dimension choice for  
softmax has been deprecated. Change the call to include dim=X as an argument.  
    return super(CategoricalAccuracyWithLogits, self).__call__(F.softmax(y_pred),  
y_true)  
Epoch 1/10: 100%|| 19/19 [02:53<00:00, 9.07s/ batches, loss=0.6694,  
acc=60.27]/usr/local/lib/python3.6/dist-  
packages/torchsample/modules/module_trainer.py:658: UserWarning: volatile was  
removed and now has no effect. Use `with torch.no_grad():` instead.  
    return Variable(input_batch, volatile=volatile), Variable(target_batch,  
volatile=volatile, requires_grad=False)  
Epoch 1/10: : 20 batches [03:37, 21.40s/ batches, loss=0.272, val_loss=0.718,  
val_acc=37.38, acc=60.27]  
Epoch 2/10: : 20 batches [03:32, 19.70s/ batches, loss=0.207, val_loss=0.77,  
val_acc=37.38, acc=65.75]  
Epoch 3/10: : 20 batches [03:34, 20.15s/ batches, loss=0.406, val_loss=0.88,  
val_acc=37.38, acc=65.75]
```

```
Epoch 4/10: : 20 batches [03:31, 19.94s/ batches, loss=0.976, val_loss=0.806,
val_acc=37.38, acc=65.75]
Epoch 5/10: : 20 batches [03:33, 20.54s/ batches, loss=0.346, val_loss=0.75,
val_acc=37.38, acc=65.75]
Epoch 6/10: : 20 batches [03:32, 19.86s/ batches, loss=0.941, val_loss=0.741,
val_acc=37.38, acc=65.75]
Epoch 7/10: : 20 batches [03:31, 19.58s/ batches, loss=0.174, val_loss=0.88,
val_acc=37.38, acc=71.23]
Epoch 8/10: : 20 batches [03:33, 19.72s/ batches, loss=0.282, val_loss=0.88,
val_acc=37.38, acc=65.75]
Epoch 9/10: : 20 batches [03:34, 20.39s/ batches, loss=0.182, val_loss=0.83,
val_acc=37.38, acc=73.97]
Epoch 10/10: : 20 batches [03:34, 20.78s/ batches, loss=0.369, val_loss=0.857,
val_acc=37.38, acc=68.49]
```

```
[0]: utils.plot_learning_curve(trainer.history)
[0]: models.calculate_roc_auc(trainer, test_loader, cuda_device)
```

### 3 Save/load model

```
[0]: torch.save(net, 'output/models/softmax-output.pt')
torch.save(net.state_dict(), 'output/models/softmax-output_state-dict.pt')
[0]: net = models.ClassificationModel3D()
net.load_state_dict(torch.load('output/models/softmax-output_state-dict.pt'))
[0]: <All keys matched successfully>
[0]: if torch.cuda.is_available():
    net.cuda()
    cuda_device = torch.cuda.current_device()
    print('Moved network to GPU')
else:
    cuda_device = -1
    print('GPU not available')
```

Moved network to GPU

### 4 K-Fold Cross Validation

```
[ ]: # This is used to report the accuracy and ROC in the paper.
# We take the metrics on the test fold after 20 epochs, regardless of how good
# they are or if they've been better during an earlier epoch.

from sklearn.model_selection import KFold
kfold = KFold(n_splits=5, shuffle=True)
```

```

patients_all = df['PTID'].unique()

normalize = True
num_epoch = 20

# Split patients into k folds.
for i, (indices_train, indices_test) in enumerate(kfold.split(patients_all)):
    print('{}-fold CV - Split {}'.format(kfold.n_splits, i+1))
    print()

    print('Preparing datasets...')
    patients_train, patients_test = patients_all[indices_train], ↴
    ↪patients_all[indices_test]
    train_dataset, test_dataset = datasets.build_datasets(df, patients_train, ↴
    ↪patients_test, normalize=normalize)
    train_loader, test_loader = datasets.build_loaders(train_dataset, ↴
    ↪test_dataset)

    print('Building model and trainer...')
    net, trainer, cuda_device = models.build_model()

    print('Starting training...')
    models.train_model(trainer, train_loader, test_loader, cuda_device, ↴
    ↪num_epoch=num_epoch)

    utils.plot_learning_curve(trainer.history)

    print('Evaluating ROC...')
    roc_auc = models.calculate_roc_auc(trainer, test_loader, cuda_device)
    print('ROC AUC:', roc_auc)

    print()
    print('*'*80)
    print()

```

# cnn\_vis

December 10, 2019

```
[13]: from __future__ import absolute_import, division, print_function, u
      →unicode_literals

%load_ext autoreload
%autoreload 2

import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd
%matplotlib inline
%config InlineBackend.figure_format = 'retina' # adapt plots for retina, u
      →displays

from IPython.core.debugger import set_trace
```

The autoreload extension is already loaded. To reload it, use:

```
%reload_ext autoreload
```

```
[14]: import os
import utils
from tqdm import tqdm_notebook
from sklearn.model_selection import train_test_split
import multiprocessing
import json
```

```
[15]: import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.autograd import Variable
from torch.utils.data import Dataset, DataLoader
```

```
[16]: import datasets, models, interpretation
```

# 1 Load model, dataset and brain area masks

```
[5]: net = models.ClassificationModel3D()  
net.load_state_dict(torch.load('output/models/softmax-output_state-dict.pt'))  
  
[6]: if torch.cuda.is_available():  
    net.cuda()  
    cuda_device = torch.cuda.current_device()  
    print('Moved network to GPU')  
else:  
    cuda_device = -1  
    print('GPU not available')
```

Moved network to GPU

```
[7]: # Important: Set model to eval before using any interpretation methods  
net.eval();  
  
[10]: # Load the data table with all 1.5 Tesla MRI scans from ADNI.  
df = datasets.load_data_table_15T()
```

Loading dataframe for /analysis/share/ADNI/ADNI\_tables/customized/DxByImgClean\_C  
ompleteAnnual2YearVisitList\_1\_5T.csv  
Found 1590 images in table  
Filtered out 2 of 1590 images because of failed preprocessing  
Filtered out 0 of 1588 images because of missing files  
Filtered out 619 of 1588 images that were MCI  
Final dataframe contains 969 images from 344 patients

```
[11]: # Patient-wise train-test-split.  
# Select a number of patients for each class, put all their images in the test  
→set  
# and all other images in the train set. This is the split that is used in the  
→paper to produce the heatmaps.  
test_patients_per_class = 30  
  
patients_AD = df[df['DX'] == 'Dementia']['PTID'].unique()  
patients_CN = df[df['DX'] == 'CN']['PTID'].unique()  
patients_CN = filter(lambda p: p not in patients_AD, patients_CN) # patients  
→that have both a CN and an AD scan should belong to the AD group  
  
patients_AD_train, patients_AD_test = train_test_split(patients_AD,  
→test_size=test_patients_per_class, random_state=0)  
patients_CN_train, patients_CN_test = train_test_split(patients_CN,  
→test_size=test_patients_per_class, random_state=0)
```

```

patients_train = np.concatenate([patients_AD_train, patients_CN_train])
patients_test = np.concatenate([patients_AD_test, patients_CN_test])

[ ]: train_dataset, test_dataset = datasets.build_datasets(df, patients_train, patients_test)

[12]: # Set up a binary mask for each brain area based on the AAL atlas.
# This is required for brain area occlusion and to determine the relevance per area.
# The AAL atlas can for example be retrieved from here: https://github.com/neurolabusc/MRICron/blob/master/templates/aal.nii.gz
brain_map = utils.load_nifti('data/aal.nii.gz')
brain_areas = np.unique(brain_map)[1:] # omit background

area_masks = []
for area in tqdm_notebook(brain_areas):
    area_mask = np.zeros_like(brain_map)
    area_mask[brain_map == area] = 1
    area_mask = utils.resize_image(area_mask, test_dataset.image_shape(), interpolation=0)
    area_masks.append(area_mask)

```

```

area_names = ['Precentral_L', 'Precentral_R', 'Frontal_Sup_L', 'Frontal_Sup_R',  

    ↳ 'Frontal_Sup_Orb_L', 'Frontal_Sup_Orb_R', 'Frontal_Mid_L', 'Frontal_Mid_R',  

    ↳ 'Frontal_Mid_Orb_L', 'Frontal_Mid_Orb_R', 'Frontal_Inf_Oper_L',  

    ↳ 'Frontal_Inf_Oper_R', 'Frontal_Inf_Tri_L', 'Frontal_Inf_Tri_R',  

    ↳ 'Frontal_Inf_Orb_L', 'Frontal_Inf_Orb_R', 'Rolandic_Oper_L',  

    ↳ 'Rolandic_Oper_R', 'Supp_Motor_Area_L', 'Supp_Motor_Area_R', 'Olfactory_L',  

    ↳ 'Olfactory_R', 'Frontal_Sup_Medial_L', 'Frontal_Sup_Medial_R',  

    ↳ 'Frontal_Med_Orb_L', 'Frontal_Med_Orb_R', 'Rectus_L', 'Rectus_R',  

    ↳ 'Insula_L', 'Insula_R', 'Cingulum_Ant_L', 'Cingulum_Ant_R',  

    ↳ 'Cingulum_Mid_L', 'Cingulum_Mid_R', 'Cingulum_Post_L', 'Cingulum_Post_R',  

    ↳ 'Hippocampus_L', 'Hippocampus_R', 'ParaHippocampal_L', 'ParaHippocampal_R',  

    ↳ 'Amygdala_L', 'Amygdala_R', 'Calcarine_L', 'Calcarine_R', 'Cuneus_L',  

    ↳ 'Cuneus_R', 'Lingual_L', 'Lingual_R', 'Occipital_Sup_L', 'Occipital_Sup_R',  

    ↳ 'Occipital_Mid_L', 'Occipital_Mid_R', 'Occipital_Inf_L', 'Occipital_Inf_R',  

    ↳ 'Fusiform_L', 'Fusiform_R', 'Postcentral_L', 'Postcentral_R',  

    ↳ 'Parietal_Sup_L', 'Parietal_Sup_R', 'Parietal_Inf_L', 'Parietal_Inf_R',  

    ↳ 'SupraMarginal_L', 'SupraMarginal_R', 'Angular_L', 'Angular_R',  

    ↳ 'Precuneus_L', 'Precuneus_R', 'Paracentral_Lobule_L',  

    ↳ 'Paracentral_Lobule_R', 'Caudate_L', 'Caudate_R', 'Putamen_L', 'Putamen_R',  

    ↳ 'Pallidum_L', 'Pallidum_R', 'Thalamus_L', 'Thalamus_R', 'Heschl_L',  

    ↳ 'Heschl_R', 'Temporal_Sup_L', 'Temporal_Sup_R', 'Temporal_Pole_Sup_L',  

    ↳ 'Temporal_Pole_Sup_R', 'Temporal_Mid_L', 'Temporal_Mid_R',  

    ↳ 'Temporal_Pole_Mid_L', 'Temporal_Pole_Mid_R', 'Temporal_Inf_L',  

    ↳ 'Temporal_Inf_R', 'Cerebellum_Crus1_L', 'Cerebellum_Crus1_R',  

    ↳ 'Cerebellum_Crus2_L', 'Cerebellum_Crus2_R', 'Cerebellum_3_L', 'Cerebellum_3_R',  

    ↳ 'Cerebellum_4_5_L', 'Cerebellum_4_5_R', 'Cerebellum_6_L', 'Cerebellum_6_R',  

    ↳ 'Cerebellum_7b_L', 'Cerebellum_7b_R', 'Cerebellum_8_L', 'Cerebellum_8_R',  

    ↳ 'Cerebellum_9_L', 'Cerebellum_9_R', 'Cerebellum_10_L', 'Cerebellum_10_R',  

    ↳ 'Vermis_1_2', 'Vermis_3', 'Vermis_4_5', 'Vermis_6', 'Vermis_7', 'Vermis_8',  

    ↳ 'Vermis_9', 'Vermis_10']

```

# Merge left and right areas.

```

merged_area_names = [name[:-2] for name in area_names[:108:2]] + area_names[108:  

    ↳ ]

```

A Jupyter Widget

```

[13]: def get_relevance_per_area(relevance_map, normalize=True):
    relevances = np.zeros(len(area_masks))
    for i, area_mask in enumerate(area_masks):
        relevances[i] = np.sum(relevance_map * area_mask)
    if normalize:
        relevances /= relevances.sum() # make all areas sum to 1

```

```

# Merge left and right areas.
merged_relevances = np.concatenate([relevances[:108].reshape(-1, 2).sum(1), ↴
                                     relevances[108:]]) ↴

return sorted(zip(merged_area_names, merged_relevances), key=lambda (a, b): ↴
              b, reverse=True)

```

## 2 Relevance heatmaps for single images

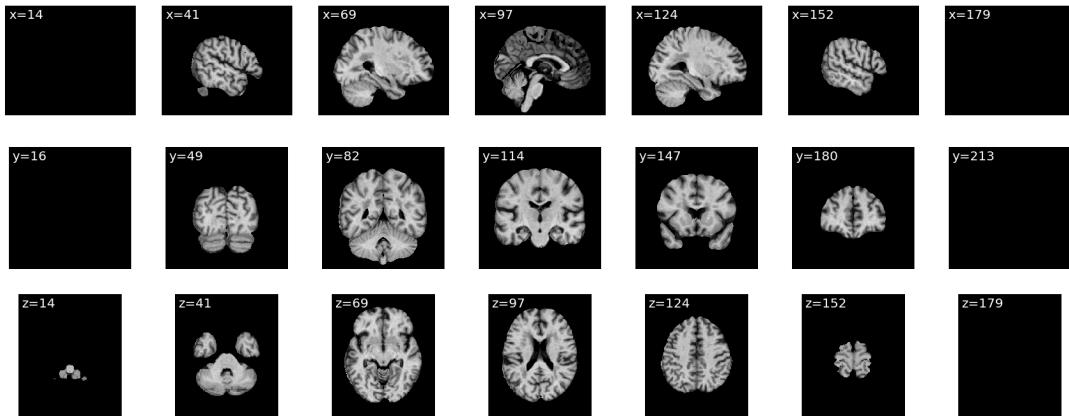
### 2.1 Raw image

```
[14]: which = 1

image_tensor = test_dataset[which][0]
raw_image = test_dataset.get_raw_image(which)

utils.plot_slices(raw_image, num_slices=7)
```

0.0 683.786425468 None None



### 2.2 Sensitivity Analysis

```
[15]: relevance_map_backprop = interpretation.sensitivity_analysis(net, image_tensor, ↴
                  cuda=True, verbose=True)
```

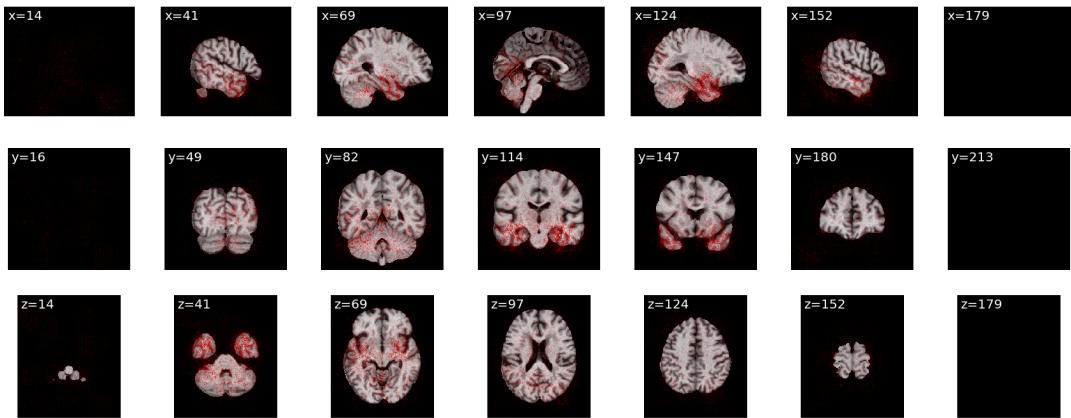
interpretation.py:62: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.

    output = F.softmax(output)

Image was classified as 1 with probability 0.942808151245

```
[16]: utils.plot_slices(raw_image, overlay=relevance_map_backprop[0], overlay_vmax=np.
    →percentile(relevance_map_backprop, 99.9), overlay_cmap=utils.
    →alpha_to_red_cmap)
```

0.0 683.786425468 0.0 0.00039409717836



```
[17]: get_relevance_per_area(relevance_map_backprop[0])[:10]
```

```
[17]: [(u'Temporal_Mid', 0.063399441580237526),
(u'Temporal_Inf', 0.057422535292876067),
(u'Fusiform', 0.045477057401692222),
(u'Cerebelum_6', 0.031064531831978527),
(u'Frontal_Mid', 0.030727307093089379),
(u'Cerebellum_Crus1', 0.030644042026538941),
(u'Lingual', 0.030407764507264345),
(u'ParaHippocampal', 0.029611374018778121),
(u'Calcarine', 0.028938094409231504),
(u'Temporal_Sup', 0.028894094165942307)]
```

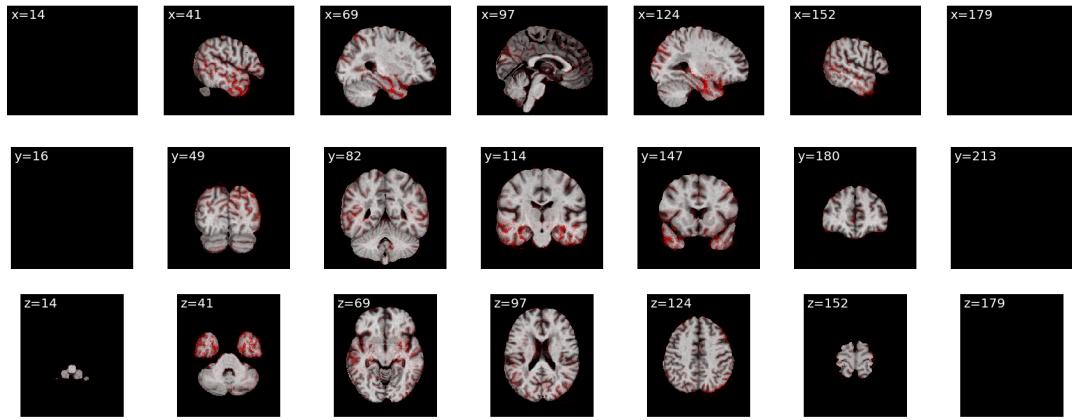
## 2.3 Guided Backpropagation

```
[18]: relevance_map_guided = interpretation.guided_backprop(net, image_tensor,
    →cuda=True, verbose=True)
```

Registered hook for layer: ReLU()  
Image was classified as 1 with probability 0.942808151245  
Removing 1 hook(s)

```
[19]: utils.plot_slices(raw_image, overlay=relevance_map_guided[0], overlay_vmax=np.
    →percentile(relevance_map_guided, 99.9))
```

```
0.0 683.786425468 0.0 8.17585628829e-05
```



```
[21]: get_relevance_per_area(relevance_map_guided[0])[:10]
```

```
[21]: [(u'Temporal_Mid', 0.092015945286800488),  
       (u'Temporal_Inf', 0.066478698797995728),  
       (u'Fusiform', 0.03954231464760289),  
       (u'Occipital_Mid', 0.038968834944943735),  
       (u'Insula', 0.038684478481913163),  
       (u'Hippocampus', 0.03862793770882194),  
       (u'Temporal_Sup', 0.037113954990888071),  
       (u'Frontal_Mid', 0.03491899978623049),  
       (u'ParaHippocampal', 0.033051167706516824),  
       (u'Temporal_Pole_Mid', 0.029052881022032715)]
```

## 2.4 Occlusion

```
[29]: relevance_map_occlusion = interpretation.occlusion(net, image_tensor, size=40,  
           stride=25, cuda=True, resize=True, verbose=True)
```

Image was classified as 1 with probability 0.942808151245

```
interpretation.py:184: UserWarning: Implicit dimension choice for softmax has  
been deprecated. Change the call to include dim=X as an argument.  
    output = F.softmax(output)
```

A Jupyter Widget

A Jupyter Widget

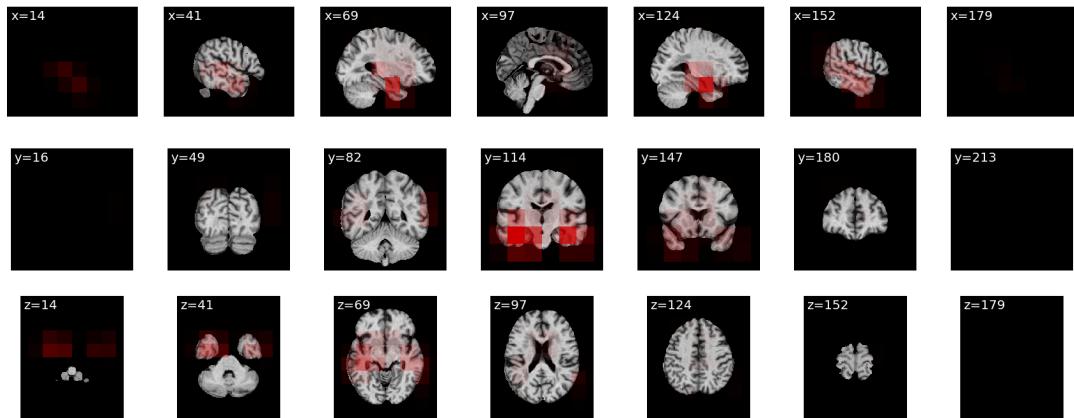
## A Jupyter Widget

interpretation.py:239: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.

```
    output = F.softmax(output)
```

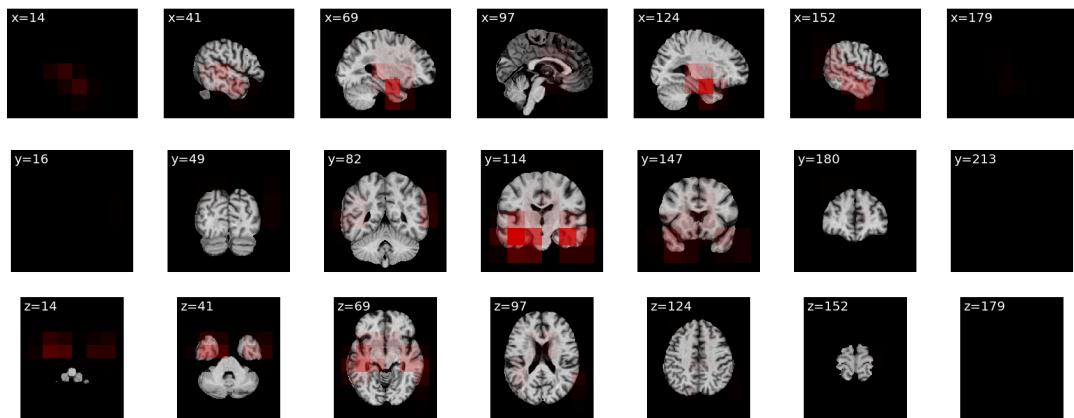
```
[30]: relevance_map_occlusion = interpretation.occlusion(net, image_tensor, size=40,  
→ stride=25, cuda=True, resize=True, verbose=True)
```

0.0 683.786425468 0.0 0.201691567898



```
[23]: utils.plot_slices(raw_image, overlay=relevance_map_occlusion,  
→ overlay_cmap=utils.alpha_to_red_cmap)
```

0.0 683.786425468 0.0 0.201691567898

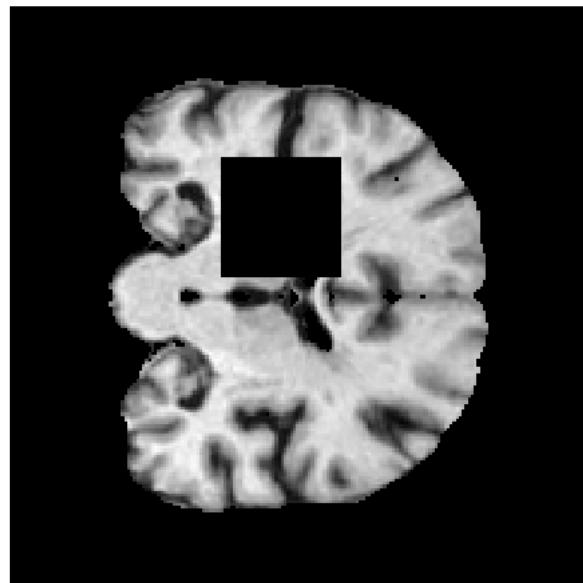


```
[24]: get_relevance_per_area(relevance_map_occlusion)[:10]
```

```
[24]: [(u'Temporal_Mid', 0.16719117220472968),  
 (u'Temporal_Sup', 0.10932730842316304),  
 (u'Temporal_Inf', 0.10875890278550929),  
 (u'Hippocampus', 0.070059933274770725),  
 (u'Fusiform', 0.060935205961371711),  
 (u'Insula', 0.060404872913250682),  
 (u'ParaHippocampal', 0.050823671802203103),  
 (u'Temporal_Pole_Sup', 0.048139119472570081),  
 (u'Putamen', 0.03351457410395263),  
 (u'Rolandic_Oper', 0.027087241700120154)]
```

```
[25]: # Plot occlusion patch on image.  
occluded_image = raw_image[:, 114, :].copy()  
size = 40  
occluded_image[50:50+size, 70:70+size] = 0  
plt.imshow(occluded_image, cmap='gray')  
plt.axis('off')
```

```
[25]: (-0.5, 192.5, 192.5, -0.5)
```



## 2.5 Area Occlusion

```
[26]: relevance_map_area_occlusion = interpretation.area_occlusion(net, image_tensor,  
→area_masks, cuda=True, verbose=True)
```

Image was classified as 1 with probability 0.942808151245

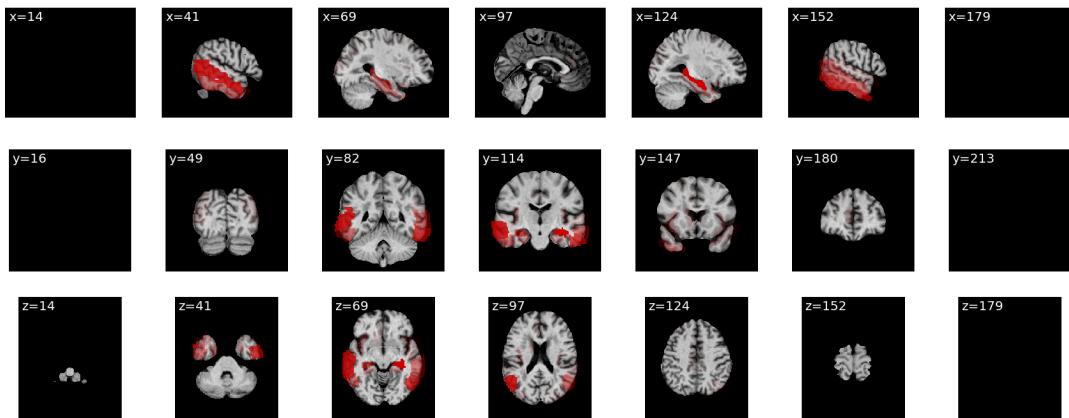
interpretation.py:290: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.  
output = F.softmax(output)

A Jupyter Widget

interpretation.py:312: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.  
output = F.softmax(output)

```
[27]: utils.plot_slices(raw_image, overlay=relevance_map_area_occlusion,  
→overlay_cmap=utils.alpha_to_red_cmap#, overlay_vmin=0, overlay_vmax=1)
```

0.0 683.786425468 0.0 0.114823



```
[28]: get_relevance_per_area(relevance_map_area_occlusion)[:10]
```

```
[28]: [(u'Temporal_Mid', 0.41732934294010604),  
(u'Temporal_Inf', 0.26899544989393337),  
(u'Hippocampus', 0.1037580938155561),  
(u'Insula', 0.039489668503406079),  
(u'ParaHippocampal', 0.03670647952330057),  
(u'Occipital_Mid', 0.023253645867581444),
```

```
(u'Temporal_Sup', 0.022765341133691098),
(u'Temporal_Pole_Mid', 0.013203760299102422),
(u'Angular', 0.0098604170389767175),
(u'Temporal_Pole_Sup', 0.0097734423833484192)]
```

### 3 Relevance heatmaps averaged over the dataset

Plot an average relevance map of all Alzheimer and all control patients (in the test set). These are the kind of plots that are included in the paper.

```
[103]: # If None, average over the entire dataset, otherwise pick a number of samples.
num_samples=None

# The background over which to plot the heatmaps.
bg = test_dataset.mean
```

#### 3.1 Backpropagation

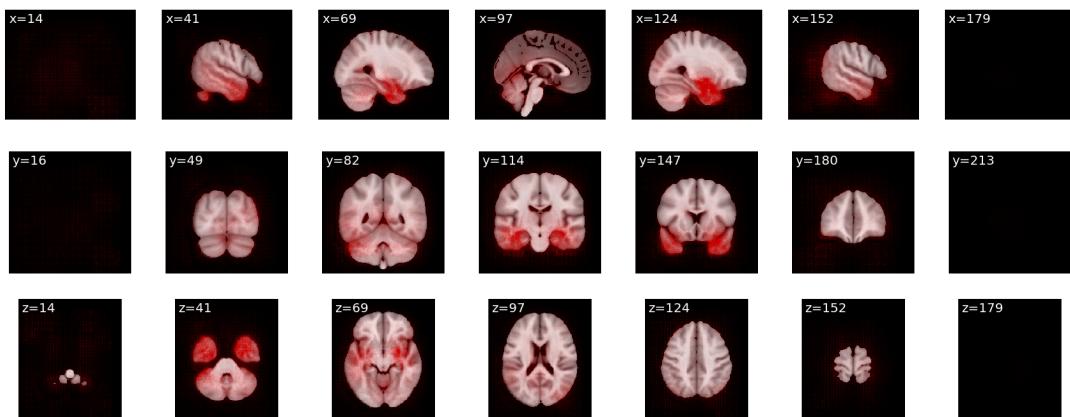
```
[101]: avg_relevance_map_AD_backprop, avg_relevance_map_NC_backprop,_
→avg_relevance_map_all_backprop = interpretation.
→average_over_dataset(interpretation.sensitivity_analysis, net, test_dataset,_
→num_samples=num_samples, seed=0, show_progress=True, cuda=True)
```

interpretation.py:64: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.  
`output = F.softmax(output)`

A Jupyter Widget

```
[104]: utils.plot_slices(bg, overlay=avg_relevance_map_AD_backprop[0], overlay_vmax=np.
→percentile(avg_relevance_map_AD_backprop, 99.9))
```

0.0 549.511221515 0.0 0.000503951129504

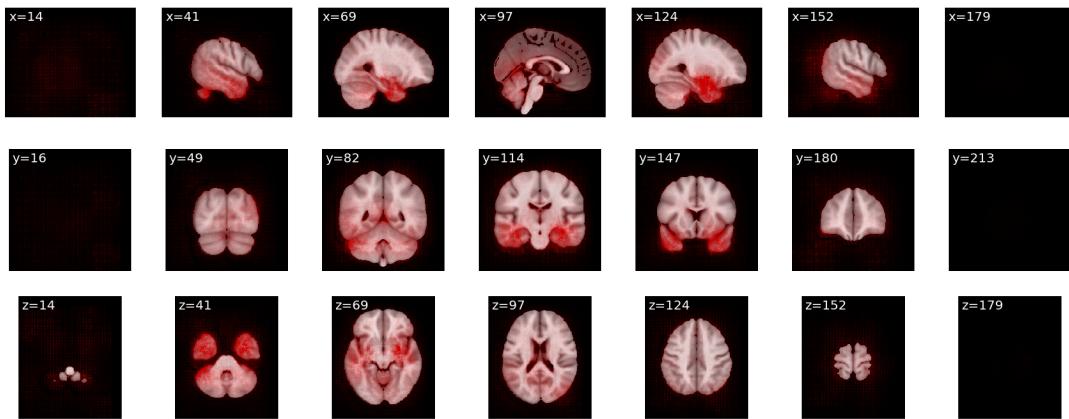


```
[105]: get_relevance_per_area(avg_relevance_map_AD_backprop)[:10]
```

```
[105]: [(u'Temporal_Mid', 0.062134743563085661),
(u'Temporal_Inf', 0.057790990228348202),
(u'Fusiform', 0.045032374647432045),
(u'Cerebellum_Crus1', 0.035149581359843862),
(u'Frontal_Mid', 0.034011630855920828),
(u'Temporal_Sup', 0.030480898464353673),
(u'Postcentral', 0.029383138050448038),
(u'Precuneus', 0.028755629744710007),
(u'Lingual', 0.028111548429315986),
(u'Cerebellum_6', 0.027862722562184854)]
```

```
[108]: utils.plot_slices(bg, overlay=avg_relevance_map_NC_backprop[0], overlay_vmax=np.
→percentile(avg_relevance_map_NC_backprop, 99.9))
```

0.0 549.511221515 0.0 0.000317961236578



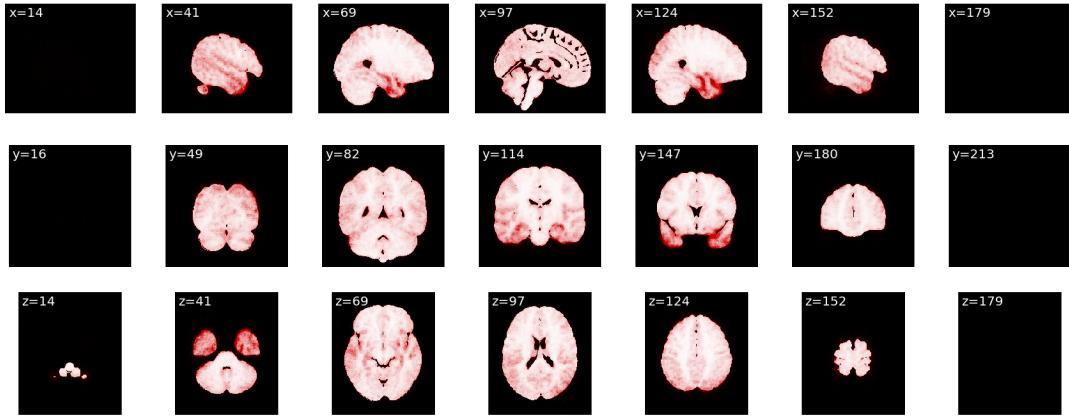
```
[109]: get_relevance_per_area(avg_relevance_map_NC_backprop)[:10]
```

```
[109]: [(u'Temporal_Mid', 0.061002153502422413),
(u'Temporal_Inf', 0.056054593440008793),
(u'Fusiform', 0.043561413191365285),
(u'Cerebellum_Crus1', 0.035870638884268641),
(u'Frontal_Mid', 0.03514580783344981),
(u'Postcentral', 0.030698061119493421),
(u'Precuneus', 0.029948179395509297),
(u'Temporal_Sup', 0.029629998754637767),
(u'Lingual', 0.028974113765431443),
(u'Cerebellum_6', 0.027856049680637488)]
```

### 3.2 Guided Backpropagation

```
[ ]: avg_relevance_map_AD_guided, avg_relevance_map_NC_guided, □  
    ↳ avg_relevance_map_all_guided = interpretation.  
    ↳ average_over_dataset(interpretation.guided_backprop, net, test_dataset, □  
    ↳ num_samples=num_samples, seed=0, show_progress=True, cuda=True)  
  
[44]: utils.plot_slices(bg, overlay=avg_relevance_map_AD_guided[0], overlay_vmax=np.  
    ↳ percentile(avg_relevance_map_AD_guided, 99.9))
```

0.0 0.999999999767 0.0 6.38190326572e-05

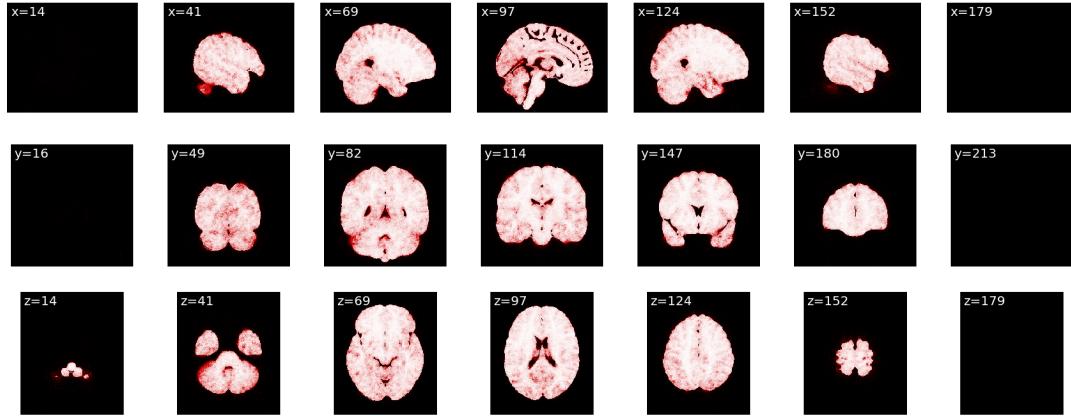


```
[115]: get_relevance_per_area(avg_relevance_map_AD_guided)[:10]
```

```
[115]: [(u'Temporal_Mid', 0.050186747294288528),  
        (u'Temporal_Inf', 0.049387297909805415),  
        (u'Cerebellum_Crus1', 0.047498072785421633),  
        (u'Frontal_Mid', 0.047045323746023951),  
        (u'Postcentral', 0.039361640925435266),  
        (u'Occipital_Mid', 0.037595296037481275),  
        (u'Parietal_Sup', 0.037276696271391033),  
        (u'Cerebellum_Crus2', 0.034529242532010909),  
        (u'Precentral', 0.033062495184460909),  
        (u'Precuneus', 0.031455642043289274)]
```

```
[159]: utils.plot_slices(bg, overlay=avg_relevance_map_NC_guided[0], overlay_vmax=np.  
    ↳ percentile(avg_relevance_map_NC_guided, 99.9))
```

0.0 0.999999999767 0.0 8.17471262417e-05



```
[160]: get_relevance_per_area(avg_relevance_map_NC_guided)[:10]
```

```
[160]: [(u'Cerebellum_Crus1', 0.049810027541261681),
(u'Temporal_Inf', 0.046468558827569076),
(u'Temporal_Mid', 0.04352493887681675),
(u'Frontal_Mid', 0.041199888656405614),
(u'Postcentral', 0.039036988259276616),
(u'Calcarine', 0.035341691163573982),
(u'Fusiform', 0.035288182948862094),
(u'Precuneus', 0.034929902892196232),
(u'Lingual', 0.032177224202014063),
(u'Occipital_Mid', 0.031029527596663088)]
```

### 3.3 Occlusion

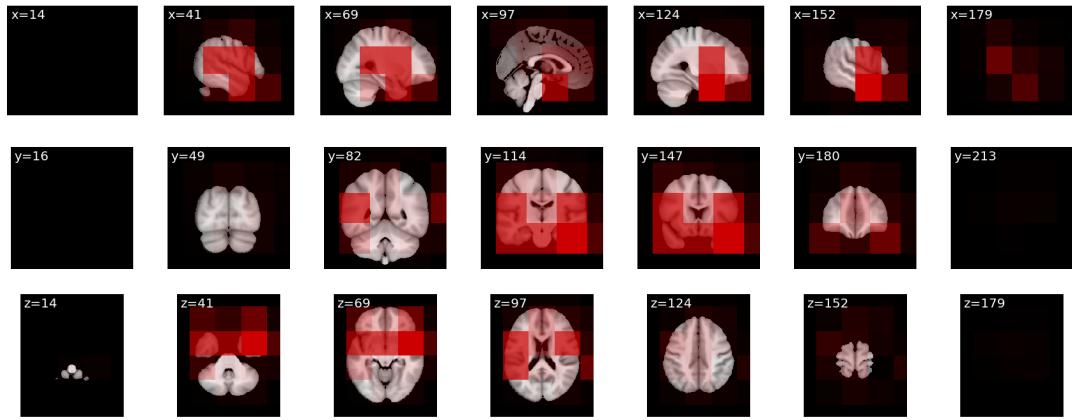
```
[115]: avg_relevance_map_AD_occlusion, avg_relevance_map_NC_occlusion, □
→avg_relevance_map_all_occlusion = interpretation.
→average_over_dataset(interpretation.occlusion, net, test_dataset, □
→num_samples=num_samples, show_progress=True, size=40, stride=40, cuda=True)
```

interpretation.py:202: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.  
 output = F.softmax(output)  
 interpretation.py:264: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.  
 output = F.softmax(output)

A Jupyter Widget

```
[116]: utils.plot_slices(bg, overlay=avg_relevance_map_AD_occlusion, □
→overlay_cmap=utils.alpha_to_red_cmap)
```

```
0.0 549.511221515 9.41125970138e-09 0.0739300968616
```

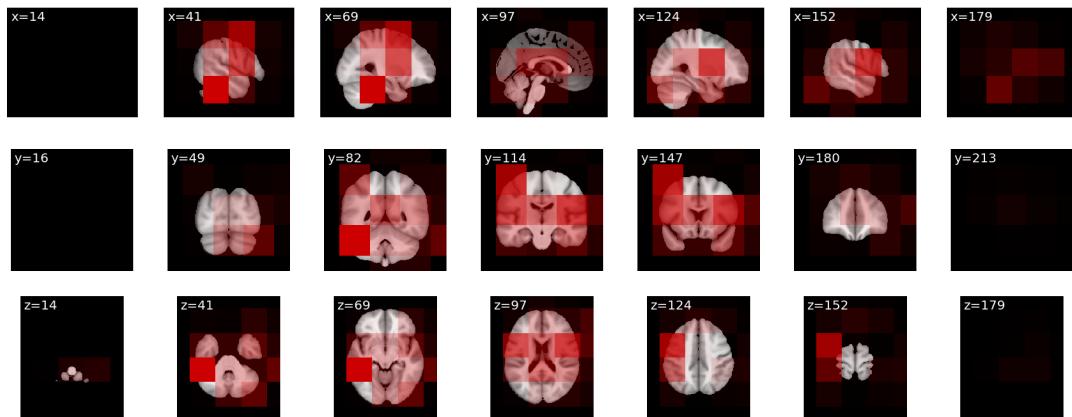


```
[117]: get_relevance_per_area(avg_relevance_map_AD_occlusion)[:10]
```

```
[117]: [(u'Temporal_Mid', 0.072957060028270862),  
 (u'Temporal_Sup', 0.0538860077993237062),  
 (u'Temporal_Inf', 0.052048787007211217),  
 (u'Insula', 0.049190853507404743),  
 (u'Postcentral', 0.040979579413481418),  
 (u'Temporal_Pole_Sup', 0.040780073290288127),  
 (u'Precentral', 0.040048014258702358),  
 (u'Frontal_Mid', 0.036851133077837384),  
 (u'Frontal_Inf_Orb', 0.034153846664753303),  
 (u'Frontal_Inf_Oper', 0.033197889095795095)]
```

```
[118]: utils.plot_slices(bg, overlay=avg_relevance_map_NC_occlusion, □  
 →overlay_cmap=utils.alpha_to_red_cmap)
```

```
0.0 549.511221515 9.63307390309e-09 0.0367510451211
```



```
[119]: get_relevance_per_area(avg_relevance_map_NC_occlusion) [:10]
```

```
[119]: [(u'Temporal_Inf', 0.061021806978649409),  
        (u'Temporal_Mid', 0.05769021858717456),  
        (u'Precentral', 0.052548755838425415),  
        (u'Postcentral', 0.051881350489086188),  
        (u'Frontal_Mid', 0.043890230859154139),  
        (u'Temporal_Sup', 0.039730211935695994),  
        (u'Fusiform', 0.039136553061625542),  
        (u'Insula', 0.036014859667420178),  
        (u'Cerebellum_Crus1', 0.034843020184713691),  
        (u'Frontal_Inf_Oper', 0.029754418467212863)]
```

### 3.4 Area Occlusion

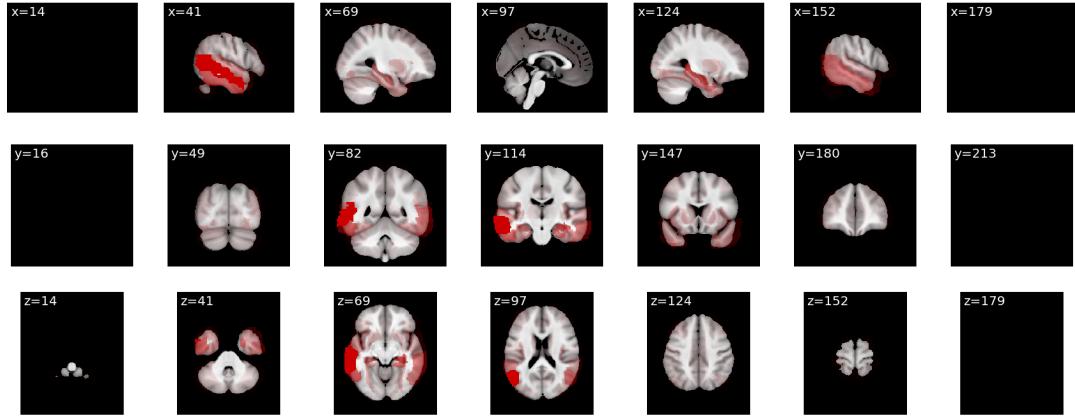
```
[129]: avg_relevance_map_AD_area_occlusion, avg_relevance_map_NC_area_occlusion,  
       ↪avg_relevance_map_all_area_occlusion = interpretation.  
       ↪average_over_dataset(interpretation.area_occlusion, net, test_dataset,  
       ↪num_samples=num_samples, seed=0, show_progress=True, cuda=True,  
       ↪area_masks=area_masks)
```

```
interpretation.py:332: UserWarning: Implicit dimension choice for softmax has  
been deprecated. Change the call to include dim=X as an argument.  
    output = F.softmax(output)  
interpretation.py:361: UserWarning: Implicit dimension choice for softmax has  
been deprecated. Change the call to include dim=X as an argument.  
    output = F.sigmoid(output)
```

A Jupyter Widget

```
[130]: utils.plot_slices(bg, overlay=avg_relevance_map_AD_area_occlusion,  
       ↪overlay_cmap=utils.alpha_to_red_cmap)
```

0.0 549.511221515 0.0 0.110895

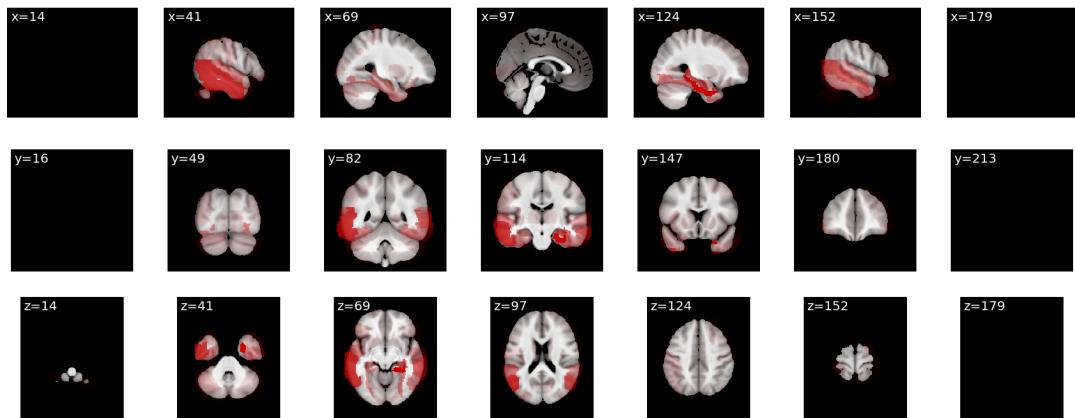


```
[131]: get_relevance_per_area(avg_relevance_map_AD_area_occlusion)[:10]
```

```
[131]: [(u'Temporal_Mid', 0.29899228606591921),
(u'Temporal_Inf', 0.12873758429466486),
(u'Temporal_Sup', 0.055523030824349881),
(u'Fusiform', 0.047355127516851739),
(u'Hippocampus', 0.039051431720235973),
(u'Insula', 0.034223421137851387),
(u'Frontal_Mid', 0.033204438981797278),
(u'ParaHippocampal', 0.032341461009729497),
(u'Occipital_Mid', 0.021591229151608399),
(u'Postcentral', 0.020695415910984304)]
```

```
[132]: utils.plot_slices(bg, overlay=avg_relevance_map_NC_area_occlusion)
```

0.0 549.511221515 0.0 0.0882691



```
[133]: get_relevance_per_area(avg_relevance_map_NC_area_occlusion)[:10]
```

```
[133]: [(u'Temporal_Mid', 0.23951053005236414),  
        (u'Temporal_Inf', 0.13724089443307996),  
        (u'Fusiform', 0.07518205198029479),  
        (u'Temporal_Sup', 0.070769070759252262),  
        (u'ParaHippocampal', 0.055645940275874196),  
        (u'Hippocampus', 0.03238929802741166),  
        (u'Postcentral', 0.032303379687998679),  
        (u'Cerebellum_Crus1', 0.032146491235223187),  
        (u'Occipital_Mid', 0.023138422637734978),  
        (u'Frontal_Inf_Orb', 0.020716323959658826)]
```

### 3.5 Save average heatmaps

```
[ ]: # Save heatmaps to file.  
np.savez_compressed('output/relevance_maps_final_compressed.npz', **{  
    'avg_relevance_map_AD_backprop': avg_relevance_map_AD_backprop,  
    'avg_relevance_map_AD_guided': avg_relevance_map_AD_guided,  
    'avg_relevance_map_AD_occlusion': avg_relevance_map_AD_occlusion,  
    'avg_relevance_map_AD_area_occlusion': avg_relevance_map_AD_area_occlusion,  
    'avg_relevance_map_AD_grad_cam': avg_relevance_map_AD_grad_cam,  
    'avg_relevance_map_NC_backprop': avg_relevance_map_NC_backprop,  
    'avg_relevance_map_NC_guided': avg_relevance_map_NC_guided,  
    'avg_relevance_map_NC_occlusion': avg_relevance_map_NC_occlusion,  
    'avg_relevance_map_NC_area_occlusion': avg_relevance_map_NC_area_occlusion,  
    'avg_relevance_map_NC_grad_cam': avg_relevance_map_NC_grad_cam  
})
```

## 4 Figures and tables for paper

```
[17]: # Load heatmaps from file.  
loaded_heatmaps = np.load('output/relevance_maps_final_compressed.npz')  
avg_relevance_map_AD_backprop = loaded_heatmaps['avg_relevance_map_AD_backprop']  
avg_relevance_map_AD_guided = loaded_heatmaps['avg_relevance_map_AD_guided']  
avg_relevance_map_AD_occlusion =  
    →loaded_heatmaps['avg_relevance_map_AD_occlusion']  
avg_relevance_map_AD_area_occlusion =  
    →loaded_heatmaps['avg_relevance_map_AD_area_occlusion']  
avg_relevance_map_AD_grad_cam = loaded_heatmaps['avg_relevance_map_AD_grad_cam']  
avg_relevance_map_NC_backprop = loaded_heatmaps['avg_relevance_map_NC_backprop']  
avg_relevance_map_NC_guided = loaded_heatmaps['avg_relevance_map_NC_guided']  
avg_relevance_map_NC_occlusion =  
    →loaded_heatmaps['avg_relevance_map_NC_occlusion']
```

```

avg_relevance_map_NC_area_occlusion =_
→loaded_heatmaps['avg_relevance_map_NC_area_occlusion']
avg_relevance_map_NC_grad_cam = loaded_heatmaps['avg_relevance_map_NC_grad_cam']

```

## 4.1 Figure 1 (Average heatmaps)

```

[16]: def plot_column(struct_arr, axes, title, cmap='gray', vmin=None, vmax=None, □
→overlay=None, overlay_cmap=utils.alpha_to_red_cmap, overlay_vmin=None, □
→overlay_vmax=None):
    if vmin is None:
        vmin = struct_arr.min()
    if vmax is None:
        vmax = struct_arr.max()
    if overlay_vmin is None and overlay is not None:
        overlay_vmin = overlay.min()
    if overlay_vmax is None and overlay is not None:
        overlay_vmax = overlay.max()
    print(vmin, vmax, overlay_vmin, overlay_vmax)

    offset = 80
    num_slices = len(axes)

    intervals = (np.asarray(struct_arr.shape) - 2 * offset) / (num_slices - 1)

    axis = 1
    axis_label = 'y'

    #for axes_column in zip(titles):

        for i, ax in enumerate(axes):
            #print(axis_label, 'plotting slice', i_slice)
            i_slice = int(np.round(offset + i * intervals[axis]))

            plt.sca(ax)
            ax.get_xaxis().set_ticks([])
            ax.get_yaxis().set_ticks([])
            #plt.axis('off')
            plt.imshow(sp.ndimage.rotate(np.take(struct_arr, i_slice, axis=axis), □
→90), vmin=vmin, vmax=vmax,
                       cmap=cmap, interpolation=None)
            plt.text(0.03, 0.97, '{}'.format(i_slice), color='white',
                    horizontalalignment='left', verticalalignment='top', □
→transform=ax.transAxes)

            if overlay is not None:

```

```

        plt.imshow(sp.ndimage.rotate(np.take(overlay, i_slice, axis=axis), ↵
         ↵90), cmap=overlay_cmap,
                    vmin=overlay_vmin, vmax=overlay_vmax, interpolation=None)

    if i == 0:
        plt.title(title + '\n')
        #plt.ylabel('Backpropagation', rotation=0, size='large')
        #ax.get_yaxis().set_label_coords(-0.5, 0.5)

```

[17]:

```

fig, axes = plt.subplots(3, 4, figsize=(11, 7.5))
axes = axes.T

#bg = mask
bg = test_dataset.mean
vmax = 550 # make the background brain a bit lighter

plot_column(bg, axes[0], 'Sensitivity Analysis\n(Backpropagation)', ↵
            ↵overlay=avg_relevance_map_NC_backprop[0], overlay_vmax=np.
            ↵percentile(avg_relevance_map_NC_backprop, 99.9), vmax=vmax)
plot_column(bg, axes[1], 'Guided\nBackpropagation', ↵
            ↵overlay=avg_relevance_map_NC_guided[0], overlay_vmax=np.
            ↵percentile(avg_relevance_map_NC_guided, 99.9), vmax=vmax)
plot_column(bg, axes[2], 'Occlusion\n', overlay=avg_relevance_map_NC_occlusion, ↵
            ↵vmax=vmax)
plot_column(bg, axes[3], 'Brain Area\nOcclusion', ↵
            ↵overlay=avg_relevance_map_NC_area_occlusion, vmax=vmax)
#plot_column(bg, axes[4], 'Grad-CAM\n', overlay=avg_relevance_map_AD_grad_cam, ↵
            ↵vmax=vmax)
plt.tight_layout()

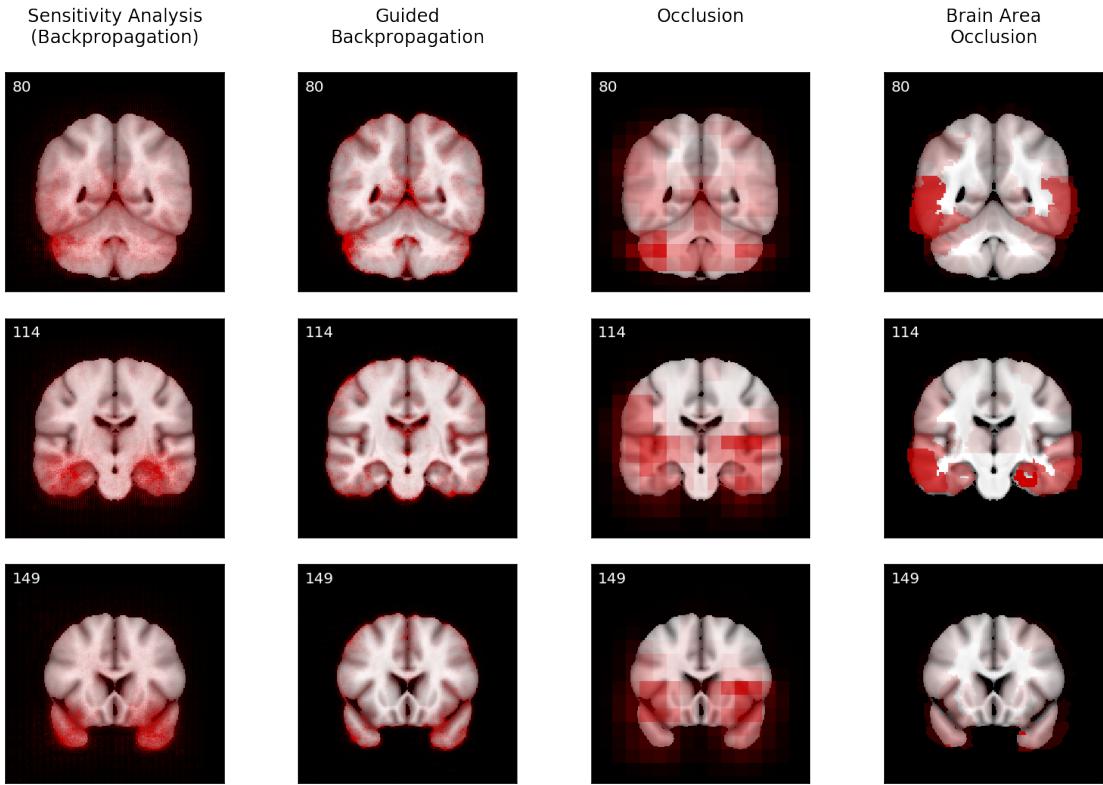
#plt.savefig('heatmaps-nc.png', dpi=300)

```

```

0.0 550 0.0 0.000433554163901
0.0 550 0.0 5.59905382397e-05
0.0 550 9.03100678415e-09 0.0754356402339
0.0 550 0.0 0.0471957

```



## 4.2 Table 1 (Most relevant brain areas)

```
[18]: most_relevant_areas_per_method = [
    get_relevance_per_area(avg_relevance_map_AD_backprop)[:4],
    get_relevance_per_area(avg_relevance_map_AD_guided)[:4],
    get_relevance_per_area(avg_relevance_map_AD_occlusion)[:4],
    get_relevance_per_area(avg_relevance_map_AD_area_occlusion)[:4] #,
    #get_relevance_per_area(avg_relevance_map_AD_grad_cam)[:4]
]

[19]: lines = []
for i in range(len(most_relevant_areas_per_method[0])):
    line = ' & '.join(['{} ({:.1f} \%)'.format(areas[i][0].replace('_', ' '), areas[i][1]*100) for areas in most_relevant_areas_per_method])
    lines.append(line)
print(' \\\n'.join(lines)) # output in latex format
```

TemporalMid (6.1 \%) & TemporalMid (7.0 \%) & TemporalMid (12.1 \%) &  
 TemporalMid (29.7 \%) \\  
 TemporalInf (5.9 \%) & TemporalInf (5.7 \%) & TemporalInf (9.2 \%) & TemporalInf  
 (14.8 \%) \\  
 Fusiform (4.6 \%) & FrontalMid (4.2 \%) & Fusiform (6.2 \%) & TemporalSup (4.4

```
\%) \\
CerebellumCrus1 (3.8 \%) & Fusiform (3.9 \%) & ParaHippocampal (5.4 \%) &
Hippocampus (4.1 \%)
```

### 4.3 Table 2 (Euclidean distances between heatmaps)

```
[20]: relevance_maps_AD = [avg_relevance_map_AD_backprop,
→ avg_relevance_map_AD_guided, avg_relevance_map_AD_occlusion,
→ avg_relevance_map_AD_area_occlusion]
relevance_maps_NC = [avg_relevance_map_NC_backprop,
→ avg_relevance_map_NC_guided, avg_relevance_map_NC_occlusion,
→ avg_relevance_map_NC_area_occlusion]
names = ['backprop ', 'guided ', 'occlusion', 'area occl']

[23]: scale_factor = 1e-4

print('Euclidean distance between average heatmaps for AD / NC samples in {}'.
→format(scale_factor))
print()

print('\t\t' + '\t'.join(names))

for a_AD, a_NC, a_name in zip(relevance_maps_AD, relevance_maps_NC, names):
    print(a_name, end='\t')
    for b_AD, b_NC, b_name in zip(relevance_maps_AD, relevance_maps_NC, names):
        print('{:.2f} / {:.2f}'.format(interpretation.heatmap_distance(a_AD,
→ b_AD) / scale_factor, interpretation.heatmap_distance(a_NC, b_NC) /
→ scale_factor), end='\t')
    print()
```

Euclidean distance between average heatmaps for AD / NC samples in 0.0001

	backprop	guided	occlusion	area occl
backprop	0.00 / 0.00	4.09 / 4.36	5.15 / 4.09	11.48 / 9.04
guided	4.09 / 4.36	0.00 / 0.00	6.47 / 5.83	11.36 / 9.80
occlusion	5.15 / 4.09	6.47 / 5.83	0.00 / 0.00	11.16 / 9.66
area occl	11.48 / 9.04	11.36 / 9.80	11.16 / 9.66	0.00 / 0.00

```
[24]: print('Euclidean distance between AD and NC heatmaps for each method in {}'.
→format(scale_factor))
print()

print('\t'.join(names))
for a, b in zip(relevance_maps_AD, relevance_maps_NC):
    print('{:.2f}'.format(interpretation.heatmap_distance(a, b) /
→ scale_factor), end='\t\t')
```

Euclidean distance between AD and NC heatmaps for each method in 0.0001

backprop            guided            occlusion            area\_occl

## 5 Export to Animation and MRIcron

```
[42]: anim = utils.animate_slices(test_dataset.mean,
    →overlay=avg_relevance_map_AD_guided[0], overlay_vmax=np.
    →percentile(avg_relevance_map_AD_guided, 99.9), axis=2,
    →reverse_direction=True, interval=70)
plt.close() # suppress plot output

[43]: plt.rcParams['animation.ffmpeg_path'] = u'/home/iliaspan/ffmpeg-3.4.
    →1-64bit-static/ffmpeg'

[44]: # Display the animation inline.
from IPython.display import HTML
HTML(anim.to_html5_video())

[44]: <IPython.core.display.HTML object>

[ ]: anim.save('data/anim-guided.gif')

[ ]: # You can also save the relevance map in NIFTI format and load it as an overlay
    →in MRIcron (https://www.nitrc.org/projects/mricron) later.
utils.save_nifti('guided_backprop.nii', relevance_map_guided[0])
```