Modern Vision Architectures Meet Alzheimer's Disease Diagnosis in sMRI

Javier Antonio Salazar Cavazos

April 10, 2024

Preface

Presentation:

- ~~Latex~~ Typst is my new best friend
- My presentation: https://typst.app/project/ruQq-xBrGHMHYNQxrlfEEN

Myself:

- Research with others[1] on Alzheimer's disease in fMRI earlier this year
- Taking EECS542 (Adv. Topics in CV) so class project is on ADD in sMRI

Some info:

- 50 million people affected (2020) [1]
- This is a hard problem due to many factors…

[1] Co-PIs: Scott Peltier & Zhongming Liu

Background[2]

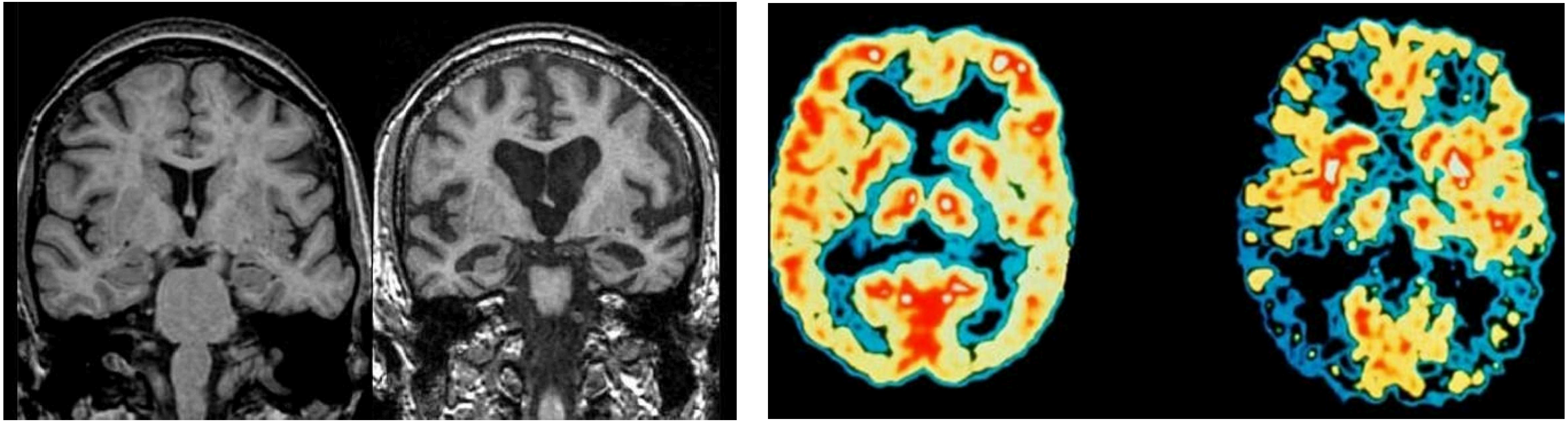


Figure 1: *sMRI and fMRI scans (left/right) on CN and AD subjects (left/right).*

[2]Figure adapted from [2]

Modern Vision Architectures

2017 *2020* *2023*

β-**VAE** [3]
Self-Supervision
"Nonlinear PCA"

Vision Transformers [4]
Self-attention based
$\ll$ inductive bias

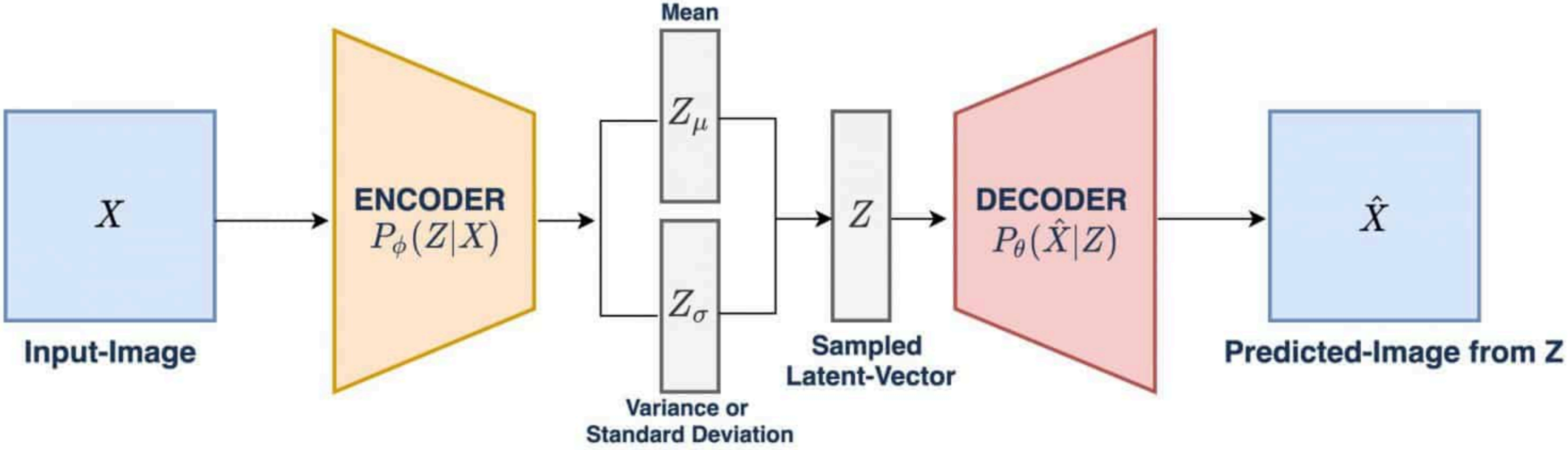
MLP-Mixer [5]
Only FC layers
Make MLPs great again!

Diffusion Models

ConvNeXt v2[3] [6]
CNN-based
Updated ResNet

[3] I implement v2 but will only discuss v1, don't worry about it

β-VAE[4]



$$Z = Z_\mu + Z_\sigma \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 1)$$

[4]Figure from [7]

β-VAE II

X = data, Z = latent variables

$P_\varphi(Z \mid X)$ = encoder, $P_\theta\left(\hat{X} \mid Z\right)$ = decoder

$$\mathcal{L}(\cdot) = \underbrace{\mathbb{E}_{P_\varphi(Z \mid X)}\left[\log\left(P_\theta\left(\hat{X} \mid Z\right)\right)\right]}_{\text{data fidelity (reconstruction)}} - \underbrace{\beta D_{KL}\left(P_\varphi(Z \mid X) \parallel \overbrace{p(Z)}^{\mathcal{N}(0,1)}\right)}_{\text{"Disentanglement" term}}$$

Vision Transformers

What are vision transformers?

Well if we ask chatGPT, we get the following:

Vision Transformers

What are vision transformers?

Well if we ask chatGPT, we get the following:

Just kidding!

Transformers I - Word Embedding

Let us tokenize[5] the following sentence: "Javier ate an apple"

Javier $\to$ 12

ate $\to$ 38

an $\to$ 5

apple $\to$ 27

D = Dictionary Length

d_k = Embedding Dim.

```
M = nn.Embed(D, d_k)
```

$$\begin{pmatrix} \longleftarrow & M(12,:) & \longrightarrow \\ \longleftarrow & M(38,:) & \longrightarrow \\ \longleftarrow & M(5,:) & \longrightarrow \\ \longleftarrow & M(27,:) & \longrightarrow \end{pmatrix}$$

sequence of tokens

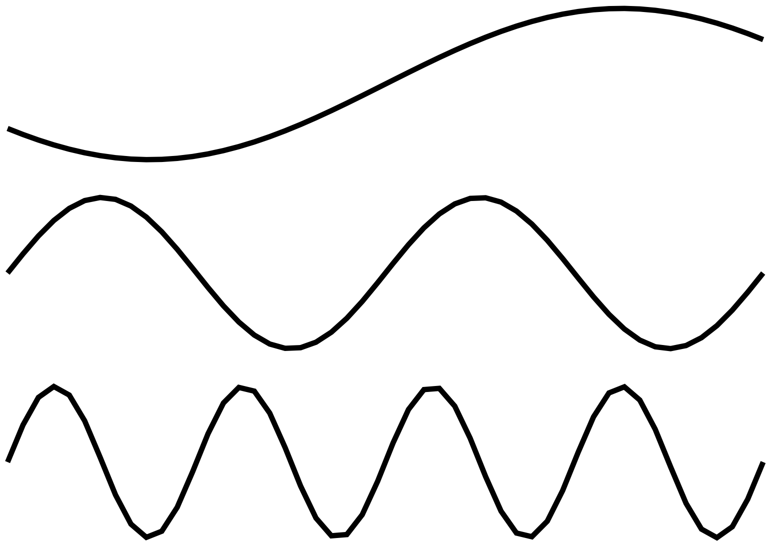
[5] Simpliest scheme is per word tokens but in practice "Huffman coding" is used here

Transformers II - Positional Encoding

Transformers are not RNNs! They don't really understand positions.

$$\begin{pmatrix} \longleftarrow & \color{blue}{\mathrm{M}(12,:)} & \longrightarrow \\ \longleftarrow & \color{red}{\mathrm{M}(38,:)} & \longrightarrow \\ \longleftarrow & \color{green}{\mathrm{M}(5,:)} & \longrightarrow \\ \longleftarrow & \color{orange}{\mathrm{M}(27,:)} & \longrightarrow \end{pmatrix}$$

T = Token matrix



Positional encoding matrix

Transformers III - Attention[6]

$$\mathrm{Attention}(Q, K, V) =$$
$$\mathrm{Softmax}\left(QK^T \div \sqrt{d_k}\right)V$$

[6] Figure from [8]. Note that this is self-attention, cross-attention is also useful.

Transformers III - Attention[7]

$$\text{Attention}(Q, K, V) =$$
$$\text{Softmax}\left(QK^T \div \sqrt{d_k}\right)V$$

T = “Javier ate an apple”

W_Q, W_K, W_V = learnable params.

$\rightarrow Q, K, V = W_Q T, W_K T, W_V T$

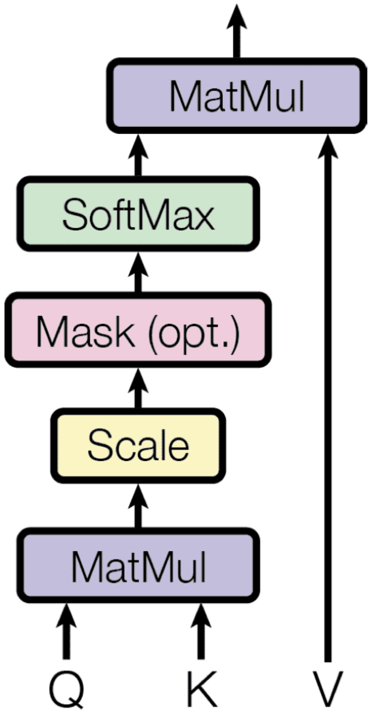
[7] Figure from [8]. Note that this is self-attention, cross-attention is also useful.

Transformers III - Attention[8]

$\text{Attention}(Q, K, V) =$
$\text{Softmax}\left(QK^T \div \sqrt{d_k}\right)V$

T = "Javier ate an apple"
W_Q, W_K, W_V = learnable params.
$\rightarrow Q, K, V = W_Q T, W_K T, W_V T$

QK^T = affinity matrix
$sm(QK^T)V$ = new embedding

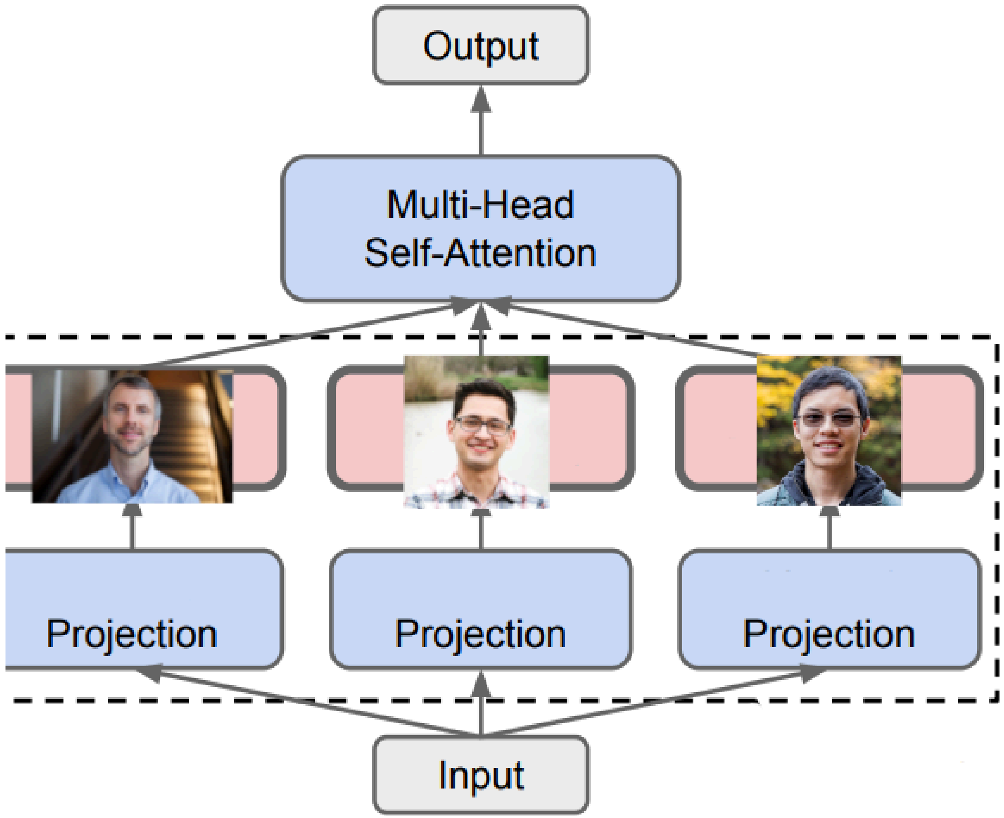


[8]Figure from [8]. Note that this is self-attention, cross-attention is also useful.

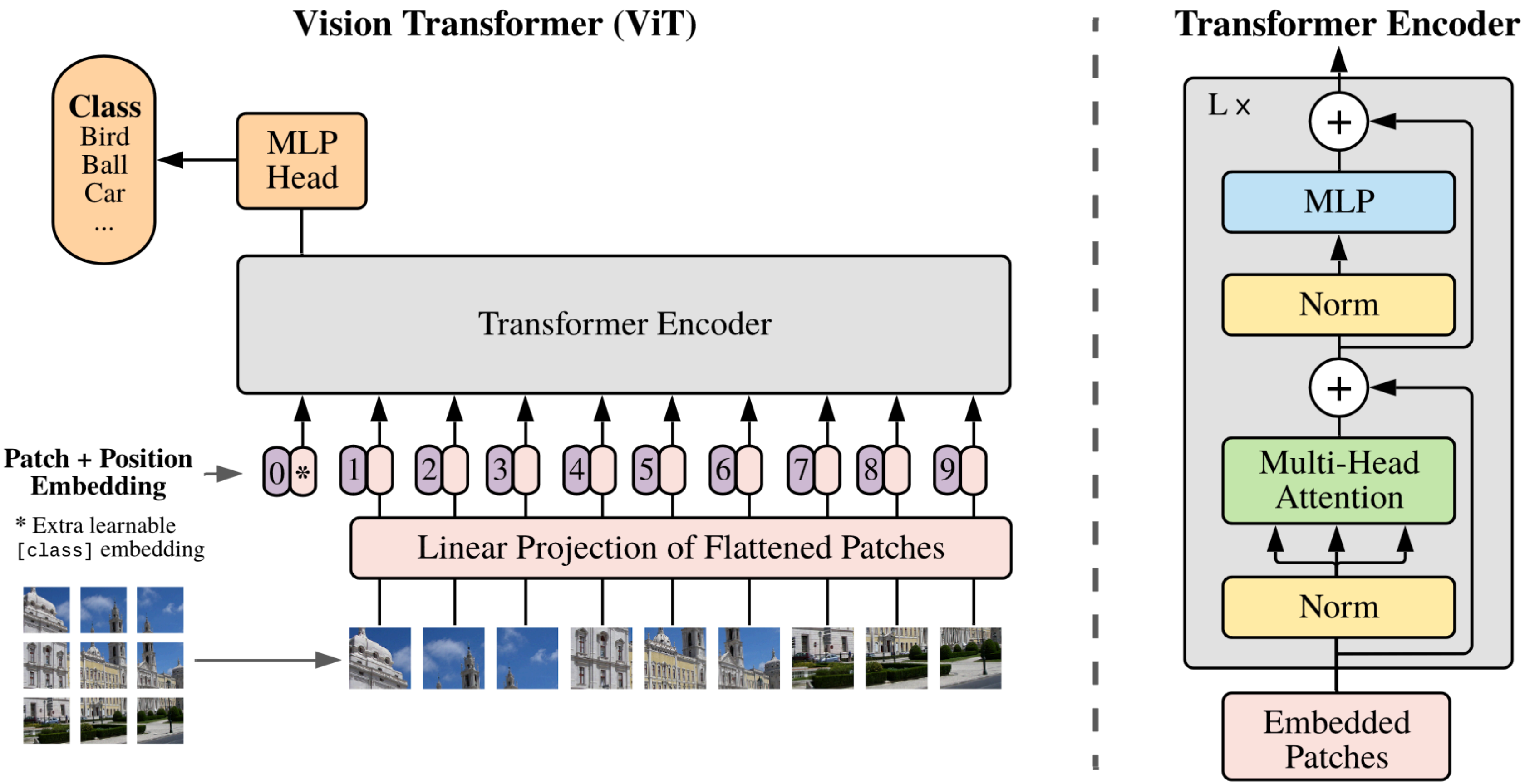
Transformers IV - Multi-Head Self-Attention

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head\_1,..., head\_3})\ W^O$$

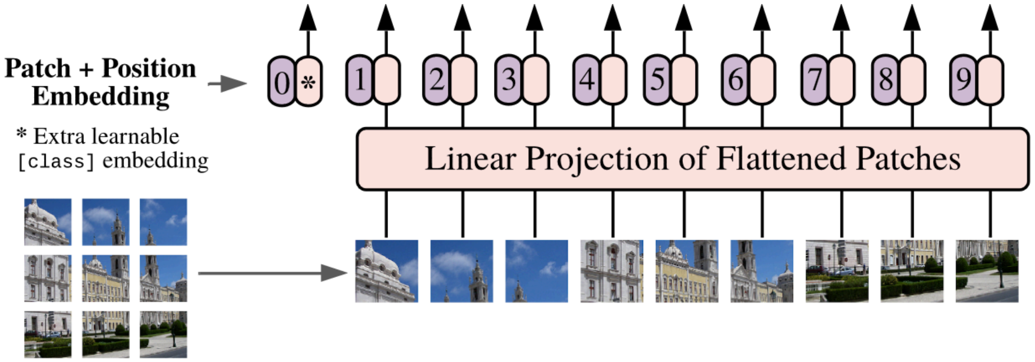
$$\overbrace{\text{head\_i}}^{\text{Jeff/Javier/...}} = \text{Attention}\left(QW_Q^i, KW_K^i, VW_V^i\right)$$



Vision Transformers I



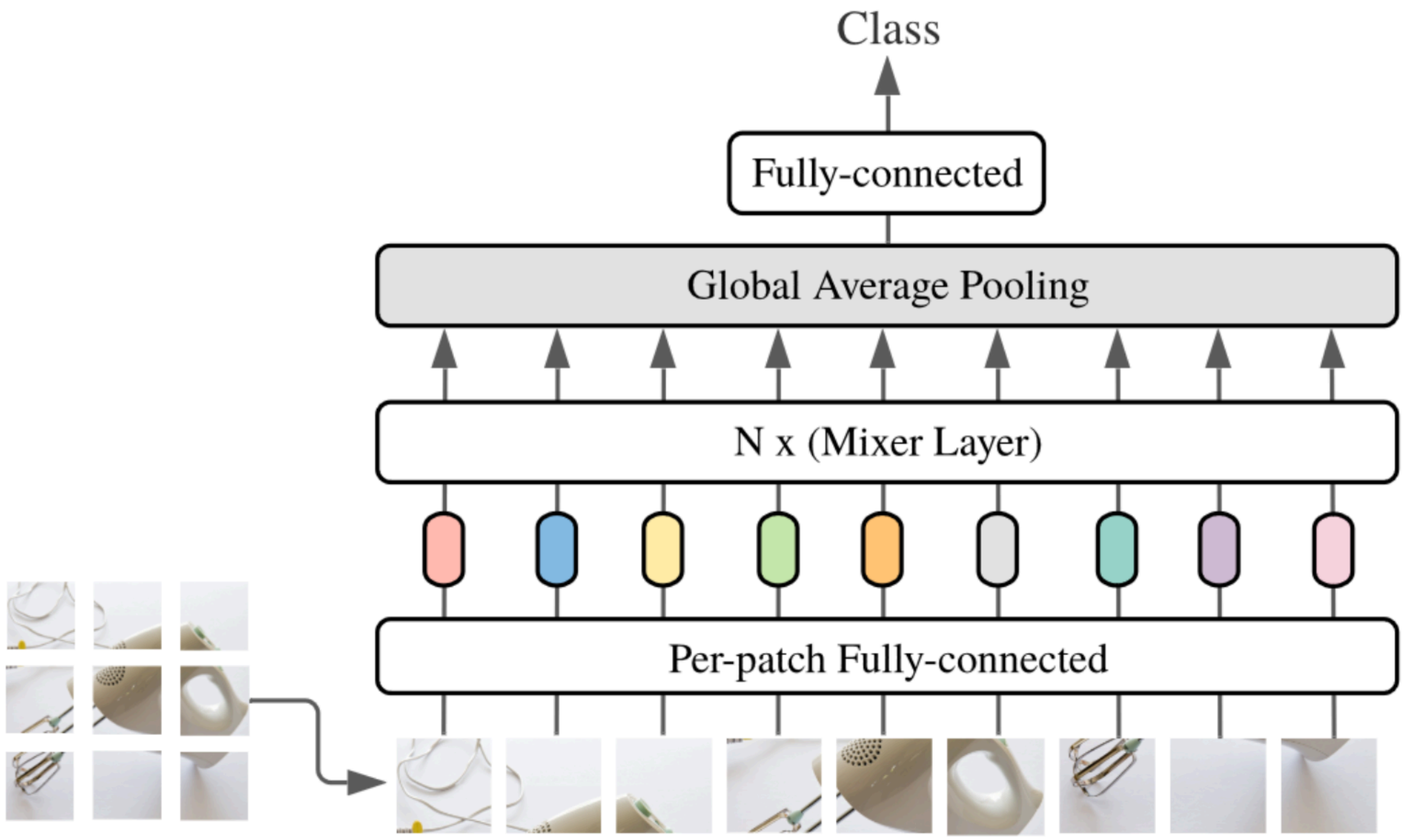
Vision Transformers II - Patch Embedding



Pytorch

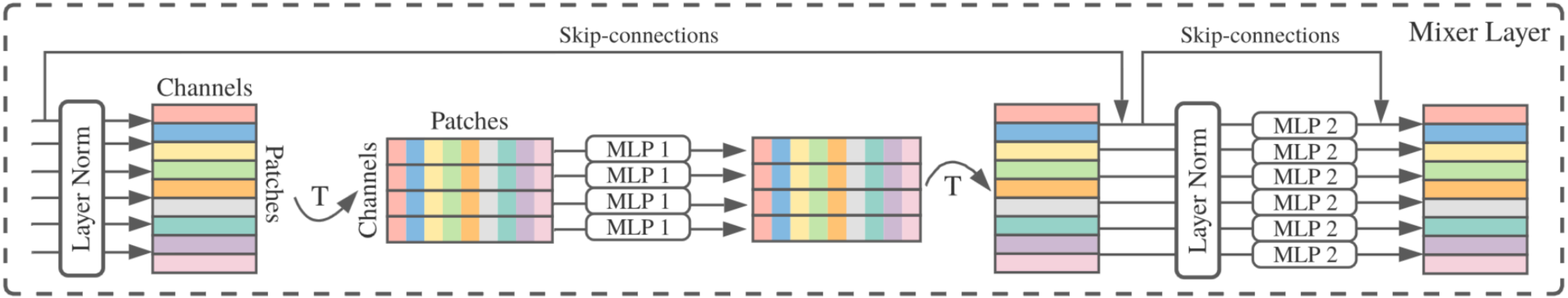
```
def __init__():
  project = nn.Conv2d(C, d_k, kernel=1, stride=patch_size)
def forward(input):
  o = project(input) # (d_k, Patch_x, Patch_y)
  o = o.flatten(2).transpose(1, 2) # (Num_patches_total, d_k)
```

MLP-Mixer


Class
Fully-connected
Global Average Pooling
N x (Mixer Layer)
Per-patch Fully-connected

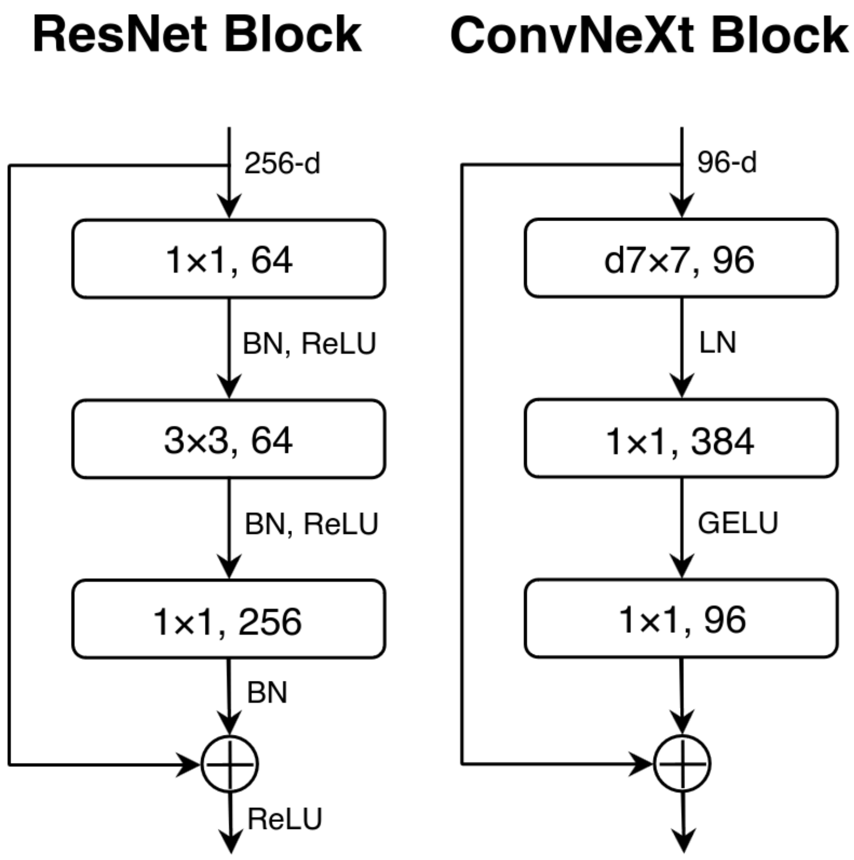
Mixer Layers

"Mixing" spatial information + channels

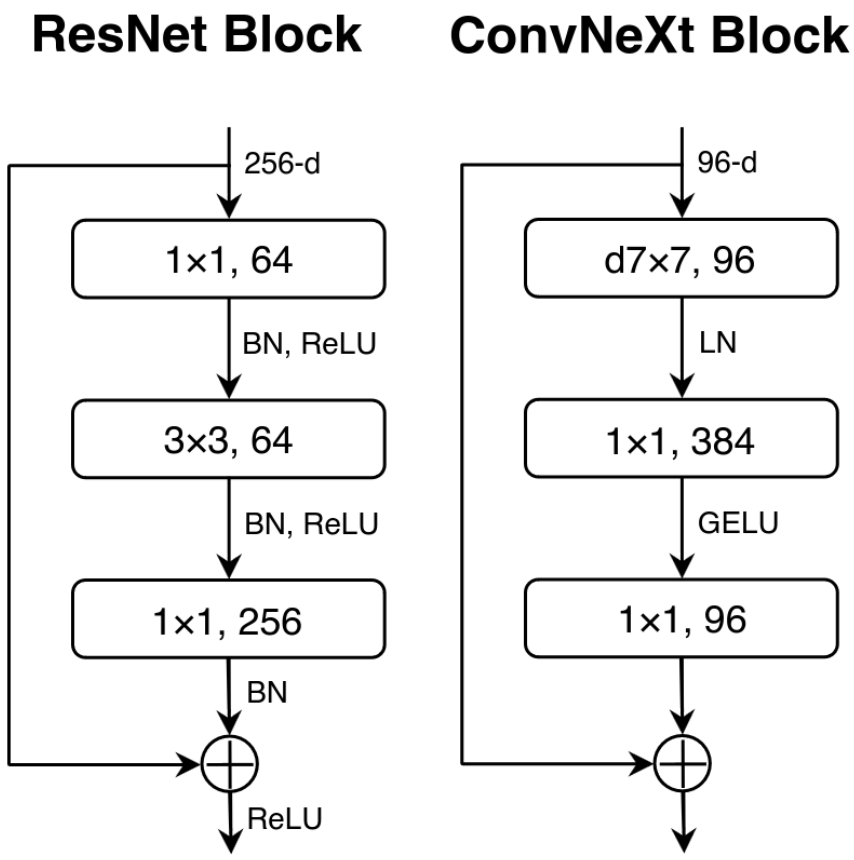


MLP → [Linear, GELU, Linear]

ConvNeXt


ResNet Block
256-d
1×1, 64
BN, ReLU
3×3, 64
BN, ReLU
1×1, 256
BN
ReLU
ConvNeXt Block
96-d
d7×7, 96
LN
1×1, 384
GELU
1×1, 96

ConvNeXt



	output size	• ResNet-50	• ConvNeXt-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4
res2	56×56	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{d}7\times7, 96 \\ 1\times1, 384 \\ 1\times1, 96 \end{bmatrix} \times 3$
res3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{d}7\times7, 192 \\ 1\times1, 768 \\ 1\times1, 192 \end{bmatrix} \times 3$
res4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{d}7\times7, 384 \\ 1\times1, 1536 \\ 1\times1, 384 \end{bmatrix} \times 9$
res5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{d}7\times7, 768 \\ 1\times1, 3072 \\ 1\times1, 768 \end{bmatrix} \times 3$
FLOPs		4.1×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6

Model Comparisons

Given some input X,

Convolution:

$$K * X = \underbrace{\mathcal{F}(K)}_{\text{weights}} \odot \mathcal{F}(X)$$

MLP:

$$\underbrace{W}_{\text{weights}} X$$

Transformer:

$$\underbrace{\mathrm{Sm}(QK^T)}_{\text{weights}} X$$

- Very different philosophies, yet very familiar
- Varying levels of inductive biases
- Unclear which approach works best given data

Moving on to Alzheimer stuff…

Datasets

Alzheimer's Disease Neuroimaging Initiative (**ADNI**):

- Develop biomarkers and advance understanding of pathophysiology
- Improve diagnostic methods for AD and improve clinical trial design
- Data: MRI, PET, genetics, cognitive tests, CSF, and blood biomarkers

Posible additional data sources:

- AIBL (sMRI/fMRI)
- OASIS (sMRI/fMRI)
- MADC (sMRI/fMRI - michigan data)
- HCP (sMRI/fMRI - healthy patients only!)

Dealing with real data is pain 🙁

Data Preprocessing

Data Distribution

Class	total (train/test)	
	Subjects	Sessions
CN	**711** (568/143)	**711** (568/143)
EMCI[9]	**326** (260/66)	**1814** (1414/401)
MCI	**388** (310/78)	**928** (760/168)
LMCI[10]	**177** (141/36)	**933** (743/190)
DAT	**276** (220/56)	**756** (612/144)

[9] This class only exists for ADNI Phase 2 / GO, overlap with MCI

[10] Same as above

Data Processing / Augmentation[11]

Pytorch

```
training_transform = tio.Compose([
    tio.ToCanonical(),
    tio.Resample(2),
    tio.CropOrPad((96, 108, 96)),
    tio.RescaleIntensity(out_min_max=(-1, 1)),
    tio.OneOf({
        tio.RandomAffine(scales=0.1, degrees=5): 1,
        tio.RandomMotion(degrees=5, translation=5): 1,
        tio.RandomNoise(std=0.05): 1, }),      ])
```

[11] TorchIO[10] library used for loading, preprocessing, augmentation of medical images

Training setup

Data:

model(sMRI $\in \mathbb{R}^{B\times 1\times 96\times 108\times 96}$) $\longrightarrow$ unnormalized logits $\in \mathbb{R}^{B\times \text{ num\_classes}}$

Optimizer:

AdamW = Adam + Weight Decay

Cost function[12]**:**

P: true distribution, Q: model distribution

$$\underbrace{H(P,Q)}_{\text{Cross Entropy}} = \underbrace{H(P)}_{\text{Entropy}} + \underbrace{D_{KL}(P \parallel Q)}_{\text{KL divergence}} = -\sum_{x\in \text{ classes}} P(x) \log Q(x)$$

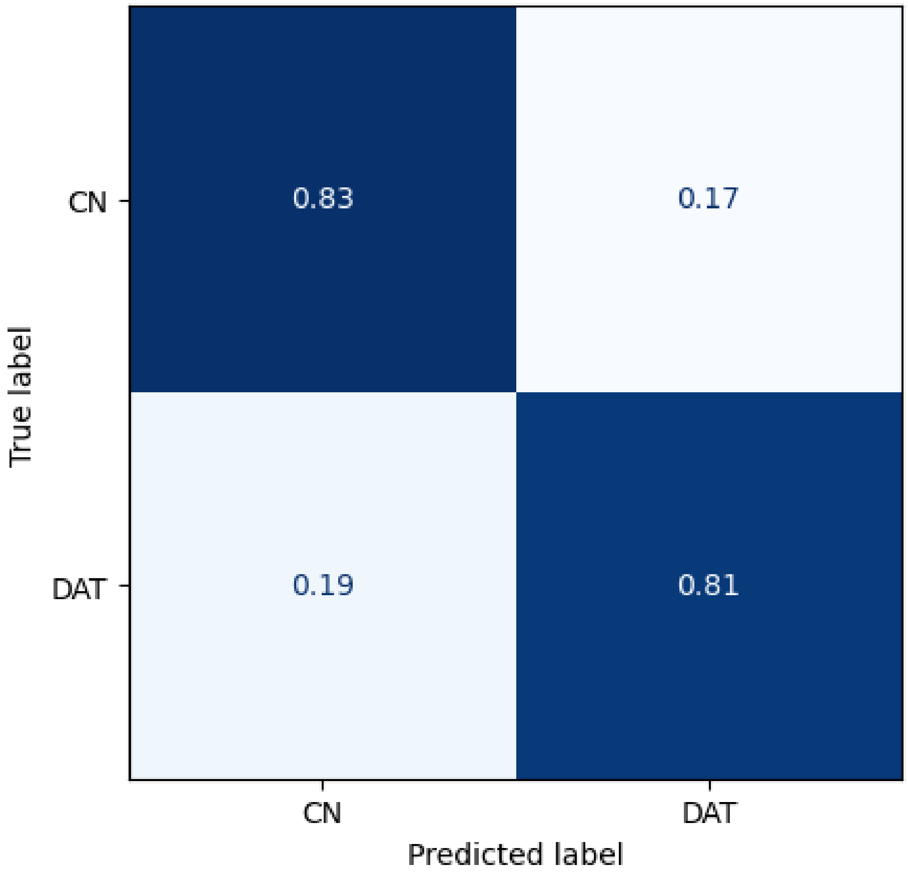
[12] I use **weighted** CE to handle class imbalances by using inverse frequencies as weights

Experiments[13]

- Model size
  - ConvNeXt ~30M vs. ~80M
- Spatial resolution
  - 1mm ($\sim 200^3$) vs. ($\sim 100^3$)
- 5-class classification
- Binary vs. 3-class classification
- Transfer learning
- Feature Maps
- VAE results

[13] Blue text indicates discussion, no additional slides

ConvNeXt - Binary Classification



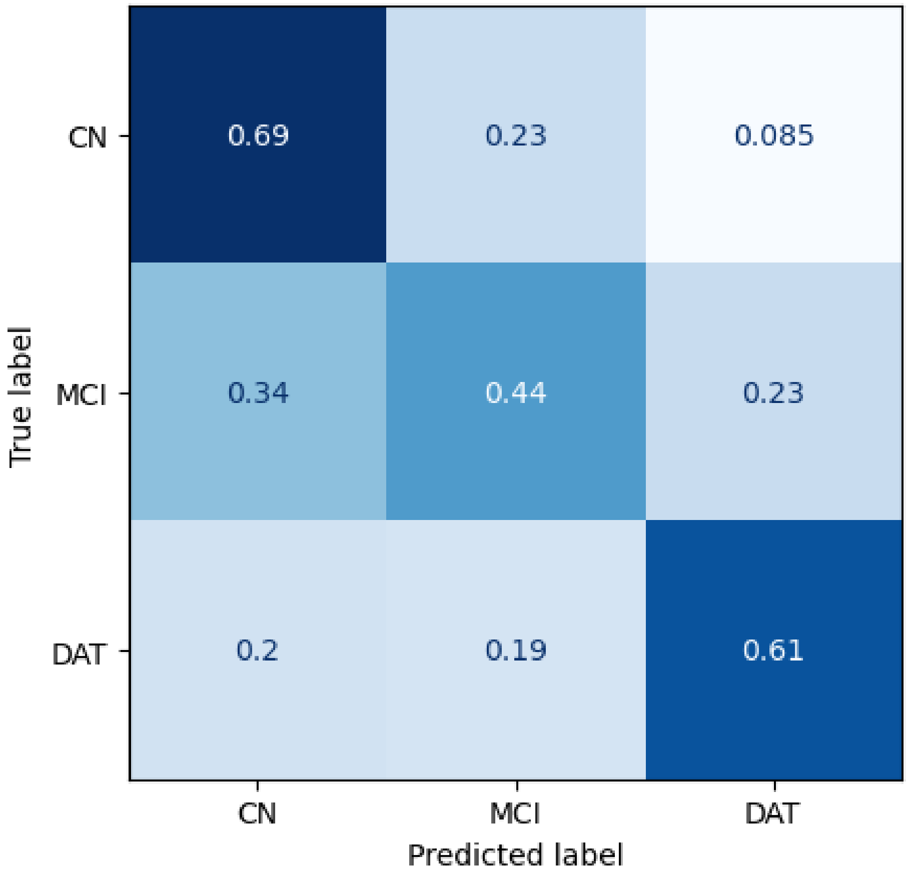
precision recall f1-score

CN
0.79 0.83 0.81

DAT
0.85 0.81 0.83

accuracy 0.82

ConvNeXt - Multi-Class Classification I

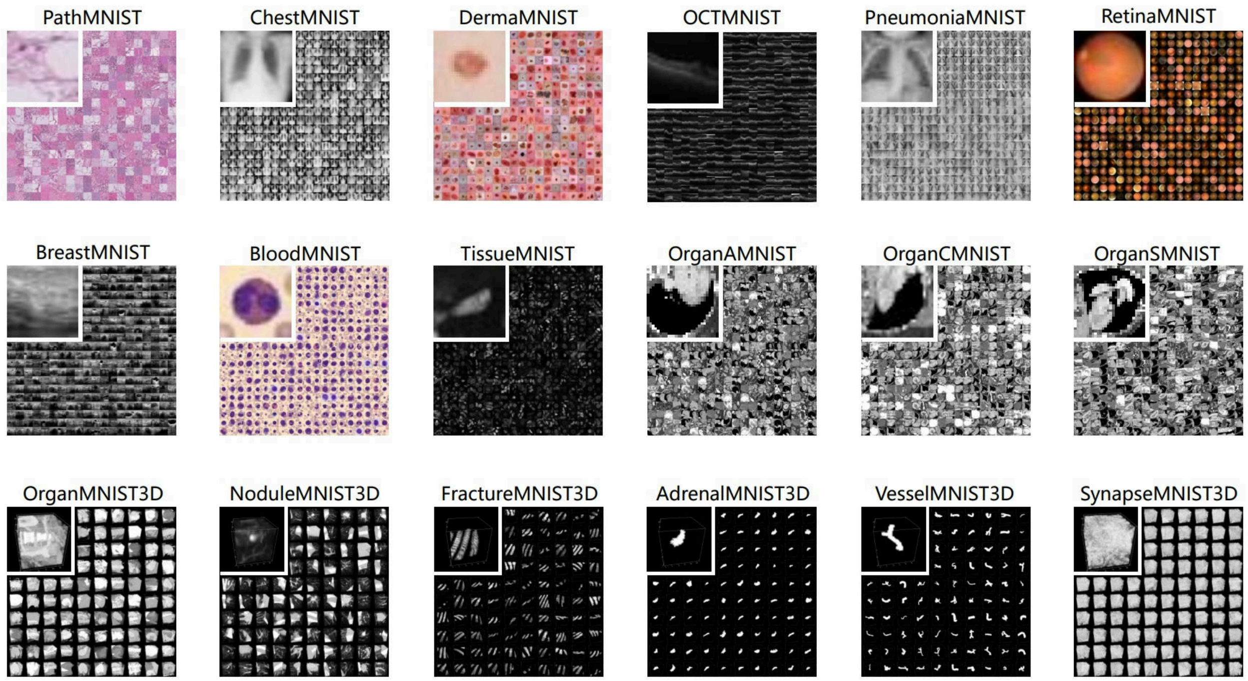


	precision	recall	f1-score
CN	0.52	0.69	0.60
MCI	0.53	0.44	0.48
DAT	0.67	0.61	0.64

accuracy 0.57

ConvNeXt - Transfer Learning

Can we use features from other problems to increase accuracy?[14]



[14] MedMNIST [11] contains 2D/3D medical data for classification from many modalities

Results

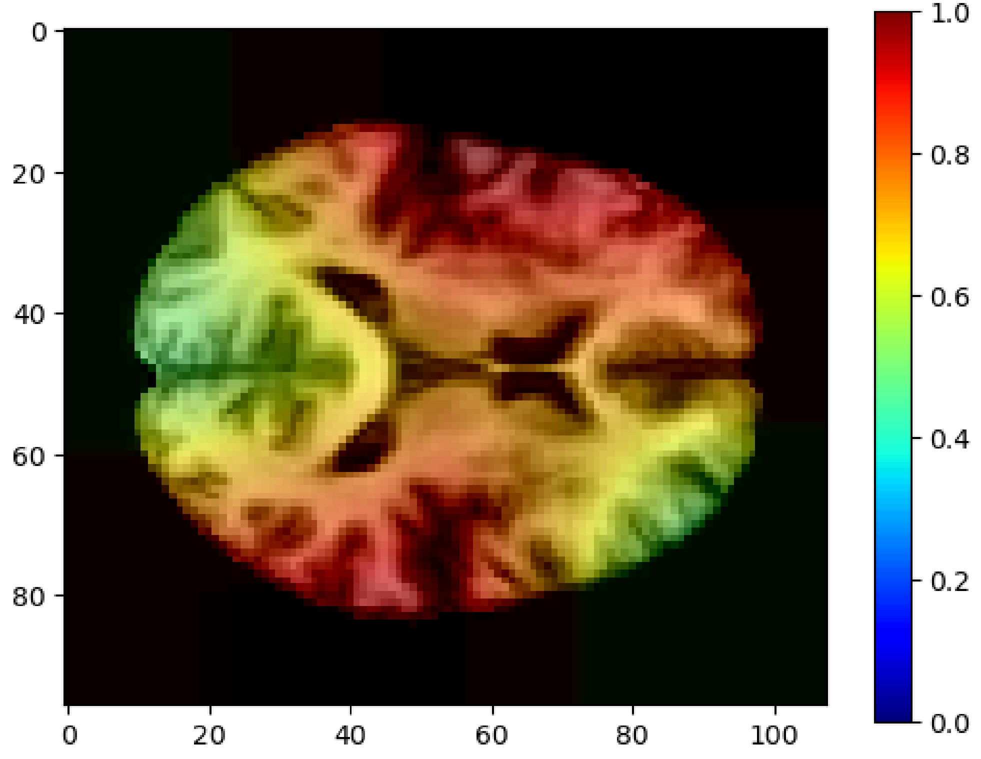
Model	Precision		Recall		Accuracy
	CN	DAT	CN	DAT	
ConvNeXt	0.79	**0.85**	**0.83**	0.81	0.82
ViT (p=8)	0.79	0.76	0.69	0.84	0.77
MLP-Mixer (p=3)	**0.87**	0.82	0.76	**0.90**	**0.84**
VAE (Self-Supervised)	0.67	0.80	0.80	0.66	0.73

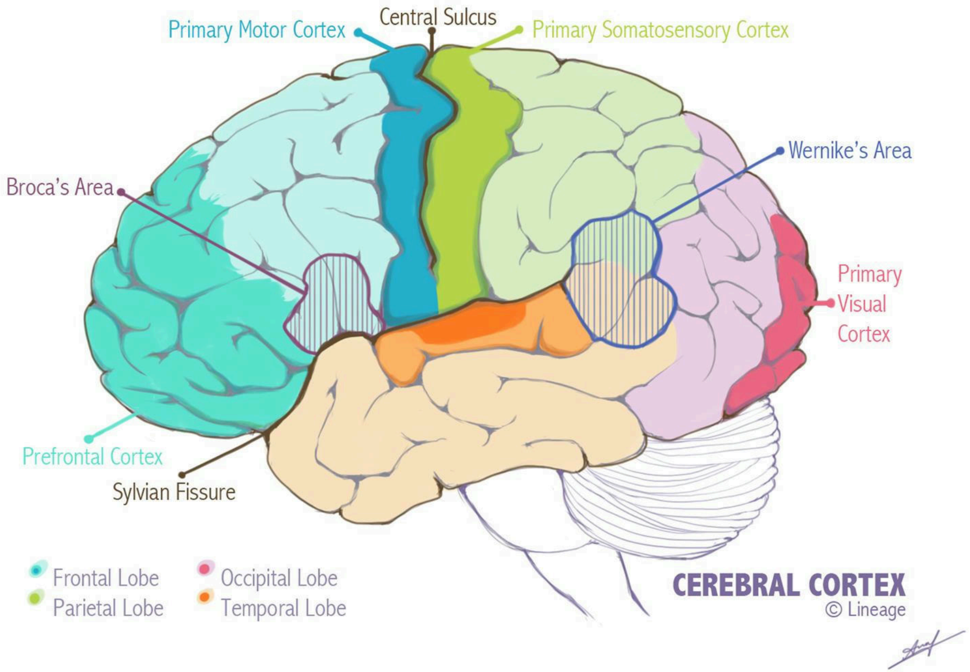
Table 1: Binary Classification Results

Model	Precision			Recall			Accuracy
	CN	MCI	DAT	CN	MCI	DAT	
ConvNeXt	0.52	**0.53**	**0.67**	**0.69**	0.44	0.61	0.57
ViT (p=8)	0.59	0.42	0.55	0.49	0.37	**0.69**	0.52
MLP-Mixer (p=3)	**0.73**	0.45	0.64	0.68	**0.53**	0.57	**0.59**
VAE (Self-Supervised)	0.47	0.45	0.57	0.64	0.40	0.46	0.49

Table 2: Three-Class Classification Results

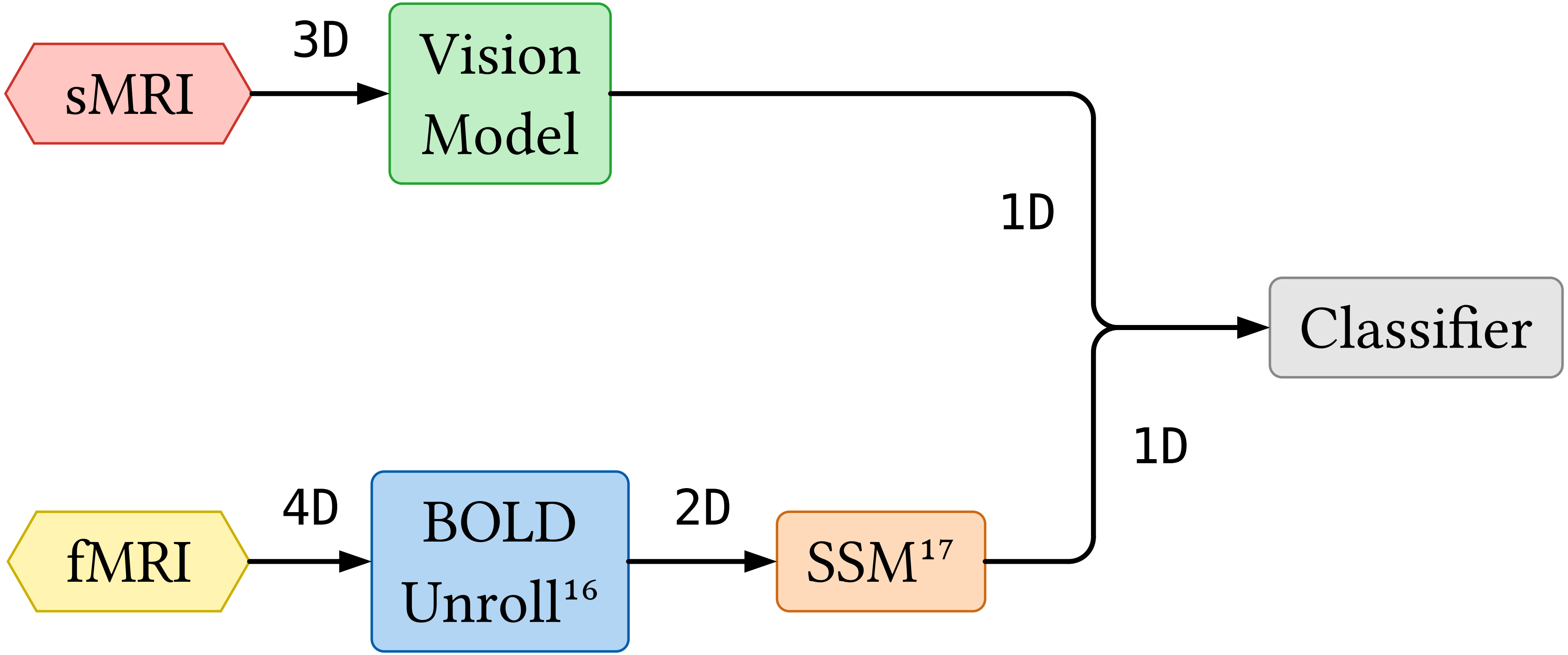
ConvNeXt - Feature Maps / Visualization[15]





[15] Grad-CAM++ method [12] used for saliency map generation

Future Work

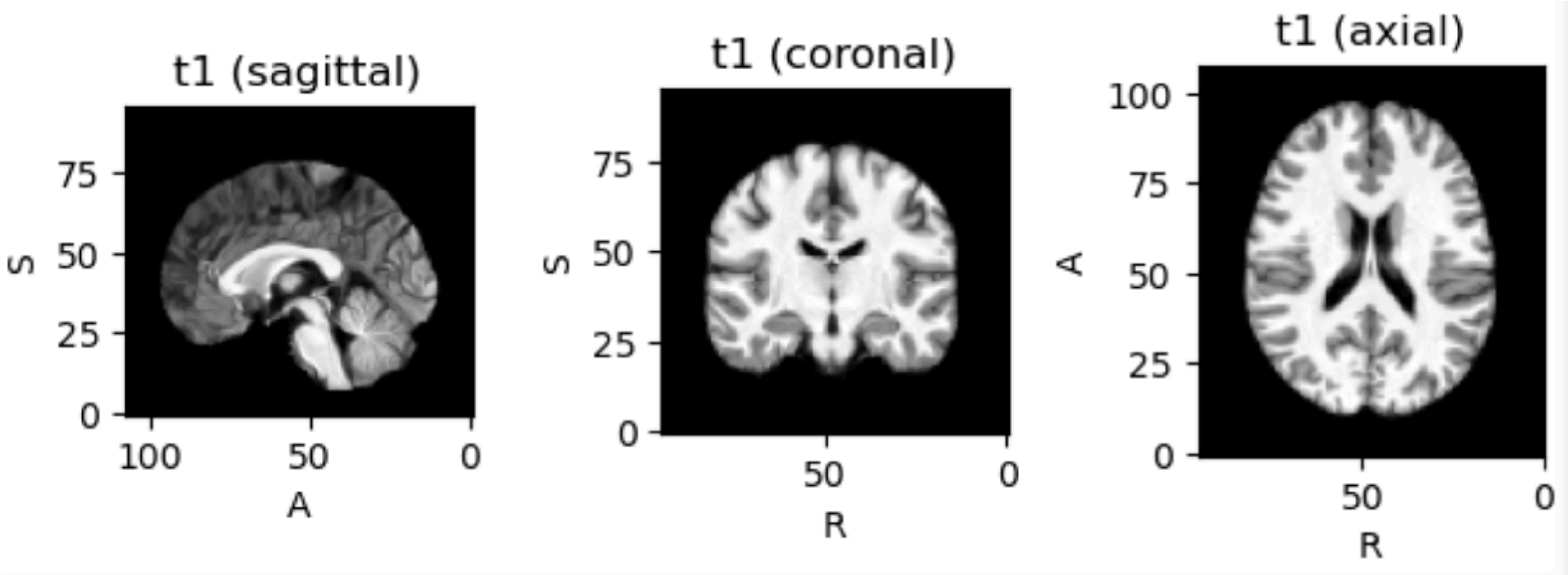


[16] (my own notation) Generate BOLD matrix w/ PE for sequence modeling

[17] More on this in the fall

Conclusion

Javier's sMRI scan[18]



[18]Thanks to Luis Hernandez-Garcia and David Frey for the scan 😃

Conclusion

$\rightarrow$ model $\rightarrow$ [**cn: 0.99**..., mci: 10^{-4}, ad: 10^{-6}]

thanks for listening!

Bibliography

[1] Z. Breijyeh and R. Karaman, “Comprehensive review on Alzheimer’s disease: causes and treatment,” *Molecules*, vol. 25, no. 24, p. 5789, 2020.

[2] E. Dinesh, M. S. Kumar, M. Vigneshwar, and T. Mohanraj, “Instinctive classification of Alzheimer's disease using FMRI, pet and SPECT images,” in *2013 7th International Conference on Intelligent Systems and Control (ISCO)*, 2013, pp. 405–409.

[3] I. Higgins *et al.*, “beta-vae: Learning basic visual concepts with a constrained variational framework.,” *ICLR (Poster)*, vol. 3, 2017.

[4] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

[5] I. O. Tolstikhin *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24261–24272, 2021.

[6] S. Woo *et al.*, "Convnext v2: Co-designing and scaling convnets with masked autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16133–16142.

[7] "https://learnopencv.com/variational-autoencoder-in-tensorflow/," vol. 0, no. , p. , 2024.

[8] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] O. Esteban *et al.*, "fMRIPrep: a robust preprocessing pipeline for functional MRI," *Nature methods*, vol. 16, no. 1, pp. 111–116, 2019.

[10] F. Pérez-García, R. Sparks, and S. Ourselin, "TorchIO: a Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning," *Computer Methods and Programs in Biomedicine*, p. 106236, 2021, doi: https://doi.org/10.1016/j.cmpb.2021.106236.

[11] J. Yang *et al.*, "Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification," *Scientific Data*, vol. 10, no. 1, p. 41, 2023.

[12] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep

convolutional networks," in *2018 IEEE winter conference on applications of computer vision (WACV)*, 2018, pp. 839–847.