Brain Tumor Detection

Evan Bosia, Igor Solomatin

Background

84,000 people are diagnosed with brain tumors ~ of which 18600 are lethal

MRI is the best technique for detecting brain tumors

MRI allows doctors to view a slice of the brain

Could benefit from automated solution

Classify MRI images as tumor or no tumor

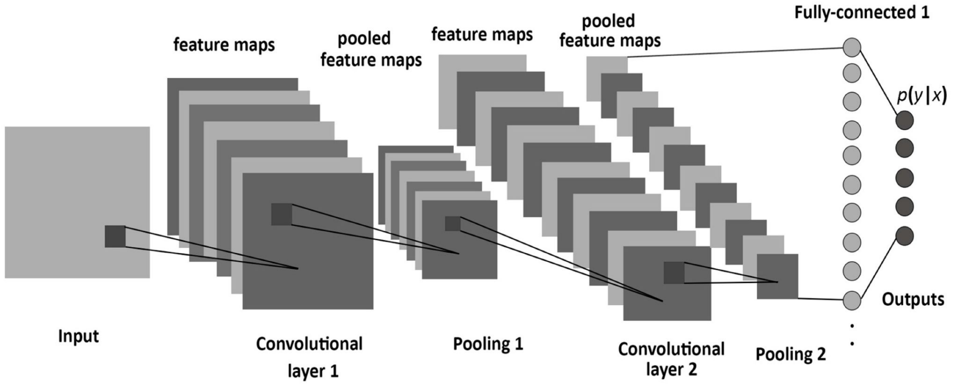
Convolution Neural Network

MRI scans are images

CNNs are great on image data

Layers:

★ Convolution + ReLU
★ Pooling with down-sampling
★ Fully Connected dense layer



https://miro.medium.com/max/4000/0*-1Pad7loK_dFOUvS.png

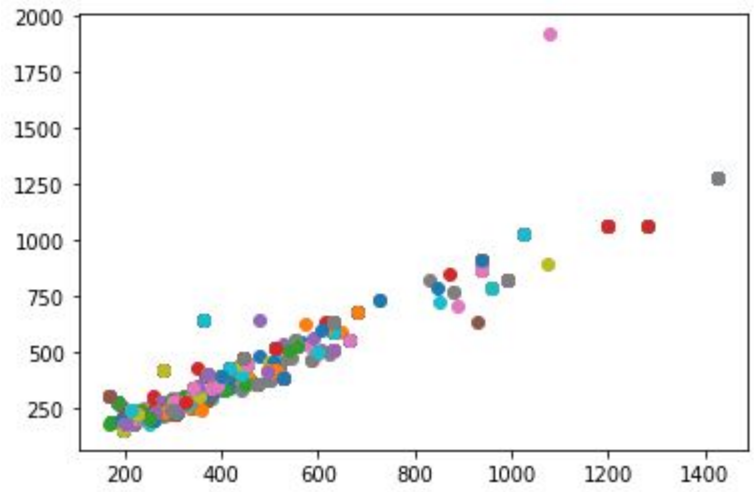
Dataset

3000 Images 50/50 split

10% of each class separated for test set

Random sizes (see distribution →)

Need fixed size images for NN

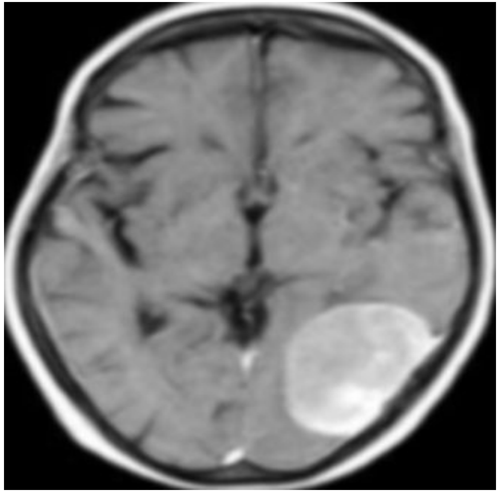


Data Preprocessing

Need to standardize image sizes

Crop brain out of images

- Gaussian blur - remove noise
- Global threshold - isolate the brain
- Find contours - segment the brain
- Crop on extreme points
- Resize crop to 224x224



Overall Model Architecture

Followed same architecture in each experiment

- Red are optional depending on experiments

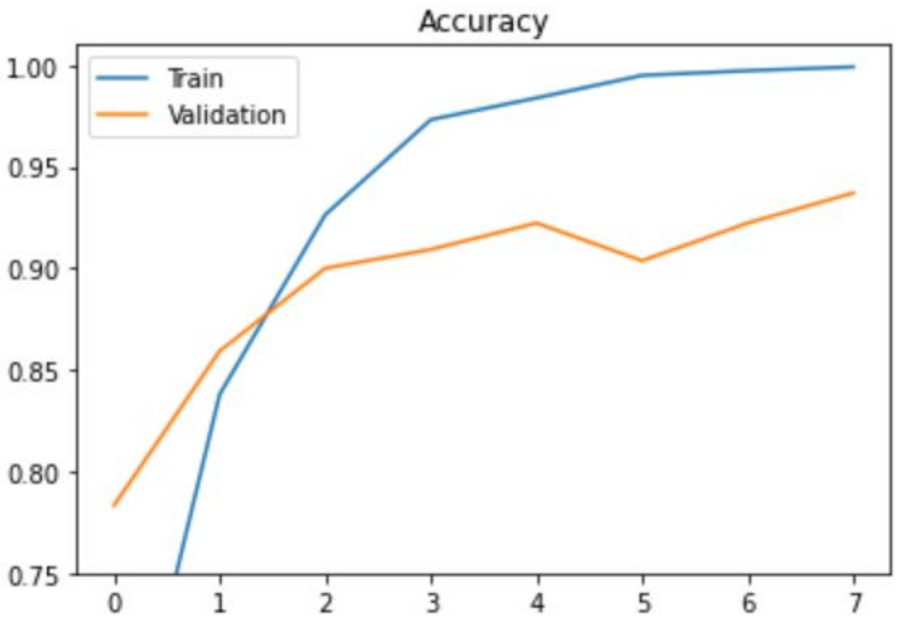
Rescale	**Data Augmentation**	**N x Convolutional**	**Dropout**	**Flatten**	**2 x Dense**
Normalize images	Transform images	Number of filters 3x3 Kernel ReLU Activation Max Pooling 2x2	20%	Image to vector	Layer Dense: 128 Layer Dense: 2

Experiment #1

One convolutional layer

- 16 filters

Max validation accuracy = 93.7%



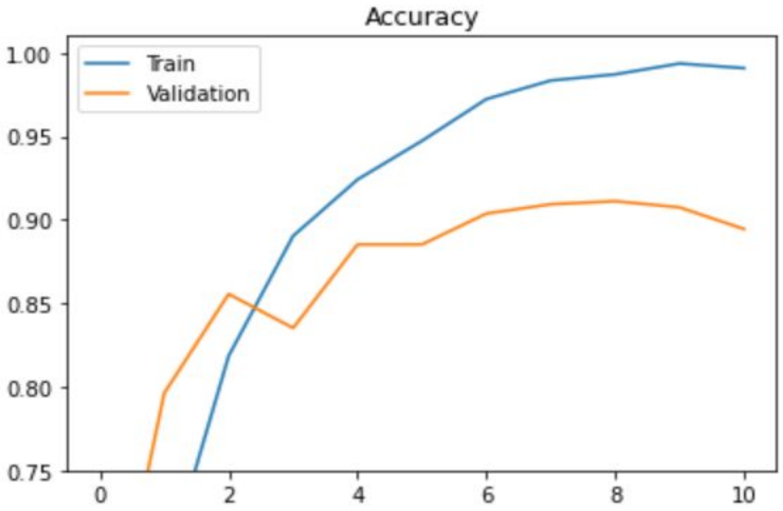
Experiment #2

Two convolutional layers

- 16 filter layer
- 32 filter layer

Add dropout 0.2

Max validation accuracy = 91.1%



Experiment #3

Three convolutional layers

- 16 filter layer
- 32 filter layer
- 64 filter layer

Still overfitting a lot!

Max validation accuracy = 91.0%

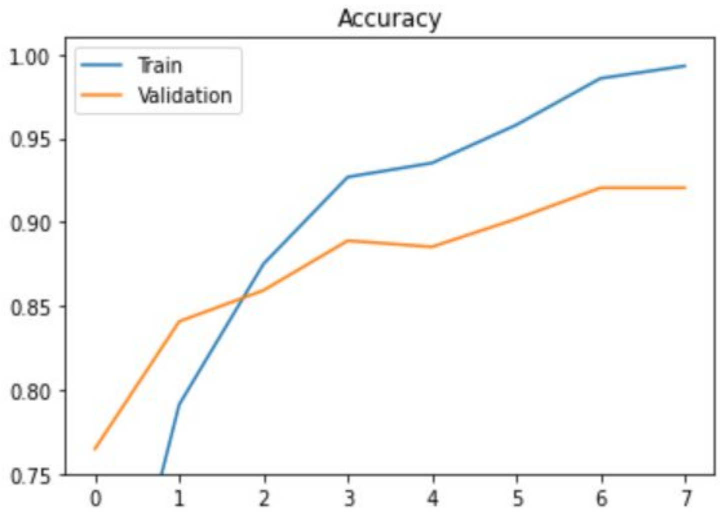


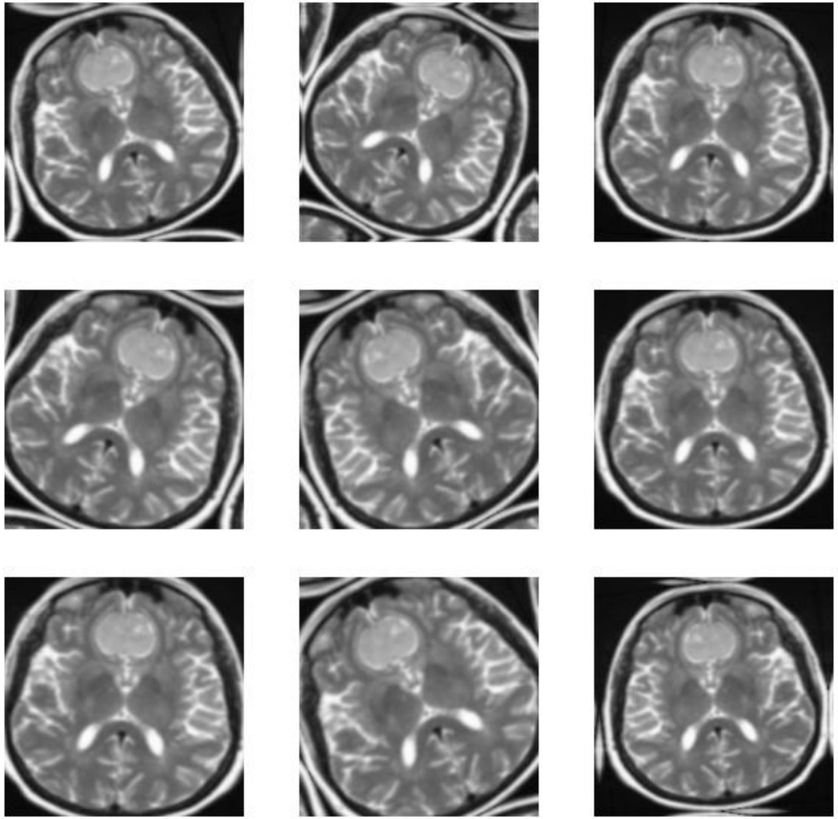
Image Augmentation

Add synthetic images with transformations

Flip / rotate / rescaling / random crop

We used flip, rotate, zoom

- No crop → do not want to remove tumor



Experiment #4

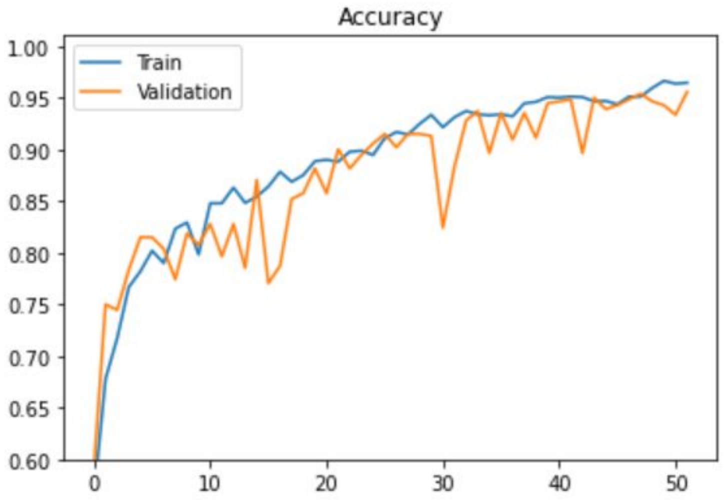
Same model as experiment #3

- Added data augmentation

Much less overfitting

Convergence is slower

Validation Accuracy: 95.6% ~ nice!



Transfer Learning

Use pretrained model as base

Add new layers on top

Freeze the base model for initial training:

- Only train the classification layers on top
- Learns off of the output of the pretrained network

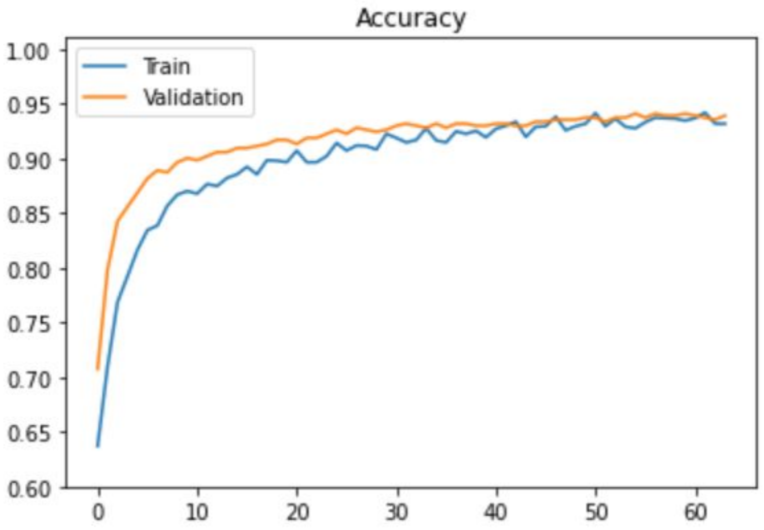
Can fine tune some number of layers afterwards for improved performance

Experiment #5

Base model: MobileNetV2

Caps at 94.1% validation accuracy

Maybe fine tuning can help?




```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
rescaling (Rescaling)        (None, 224, 224, 3)       0
_________________________________________________________________
sequential (Sequential)      (None, 224, 224, 3)       0
_________________________________________________________________
mobilenetv2_1.00_224 (Functi (None, 7, 7, 1280)        2257984
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 2562
=================================================================
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
```

Experiment #5 cont.

Fine tuning

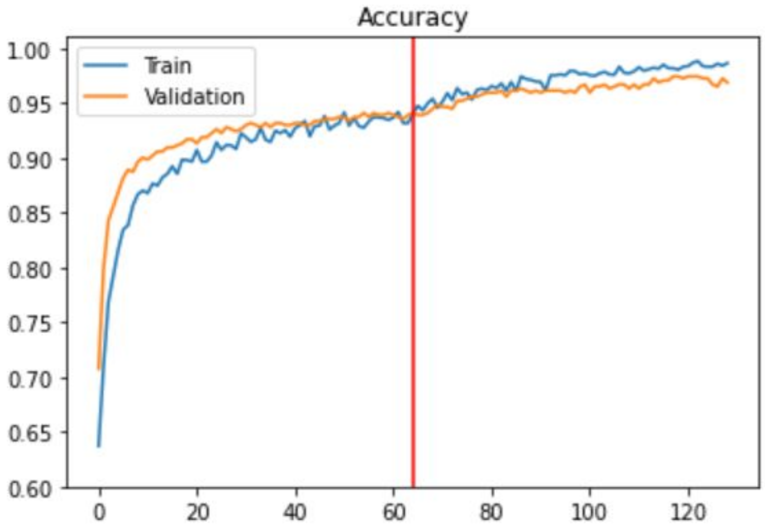
- Opens base model to training

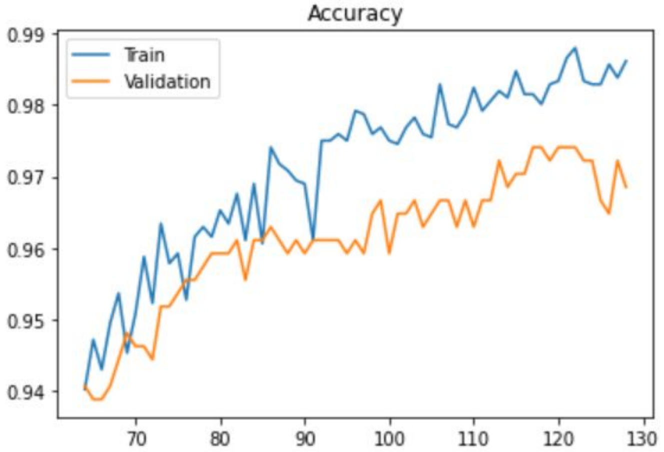
Allow training on last 50 layers

Low learning rate

Starts overfitting at the end

Best validation score: 97.4%





Regularization

L1 regularization can help with overfitting

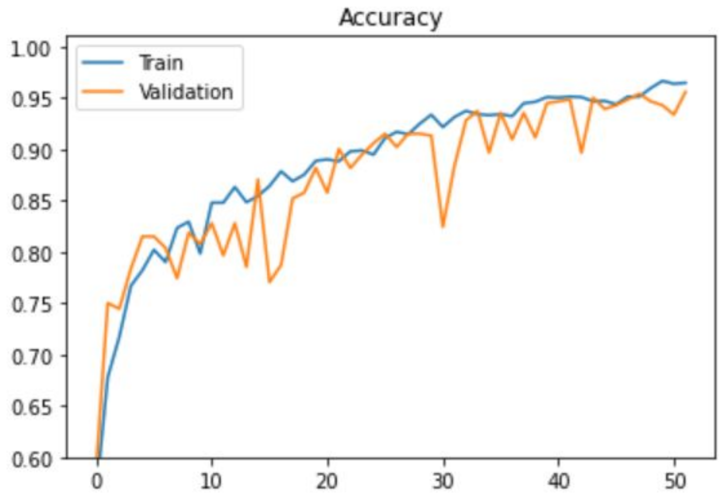
Tried on experiments 4 (top) and 5 (bottom)

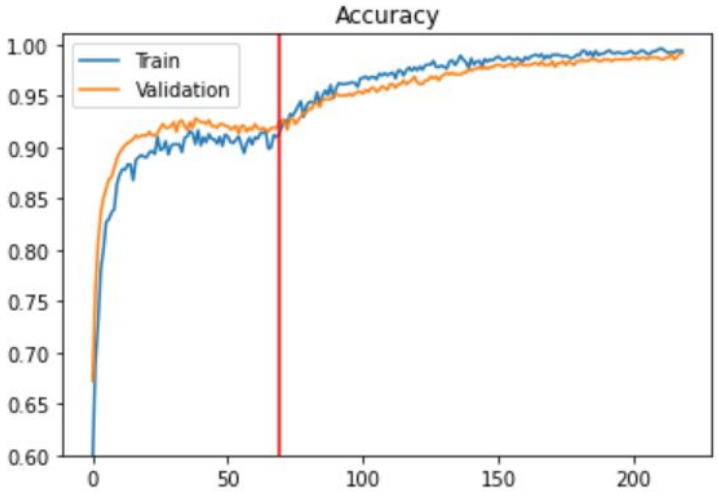
Experiment 4 had minor improvement

- Original model was not overfit

Experiment 5 had significant improvement

- 99.1% validation accuracy!





Results Comparison

All models are roughly the same on the test set (except last)

Best model is transfer learning fine-tuned with regularization ~ 99.7% accurate

name	size	layers	accuracy	true_pos	true_neg	false_pos	false_neg	pos_recall	neg_recall	pos_precision	neg_precision	pos_f1-score	neg_f1-score
experiment_001	137029	8	0.946667	138	146	4	12	0.920000	0.973333	0.971831	0.924051	0.945205	0.948052
experiment_002	148884	9	0.946667	140	144	6	10	0.933333	0.960000	0.958904	0.935065	0.945946	0.947368
experiment_003	177253	11	0.946667	143	141	9	7	0.953333	0.940000	0.940789	0.952703	0.947020	0.946309
experiment_004	483616	12	0.950000	145	140	10	5	0.966667	0.933333	0.935484	0.965517	0.950820	0.949153
experiment_004r	494670	12	0.976667	145	148	2	5	0.966667	0.986667	0.986395	0.967320	0.976431	0.976898
experiment_005a	4024888	7	0.956667	144	143	7	6	0.960000	0.953333	0.953642	0.959732	0.956811	0.956522
experiment_005b	4165246	7	0.986667	149	147	3	1	0.993333	0.980000	0.980263	0.993243	0.986755	0.986577
experiment_005ra	4035345	7	0.936667	141	140	10	9	0.940000	0.933333	0.933775	0.939597	0.936877	0.936455
experiment_005rb	4180621	7	0.996667	149	150	0	1	0.993333	1.000000	1.000000	0.993377	0.996656	0.996678

Discussion

Did not consider weighting positives and negatives

- A false negative is much worse than a false positive

Need model interpretability

- Doctors should know the reasons behind classification

The best model would be one with high negative precision and interpretability

Conclusion

Adding layers does not always help

Data augmentation helps with overfitting

Transfer learning is effective with small datasets

Regularization helps with overfitting

Questions?