fMRI and sMRI schizophrenia diagnosis classification

NeuroML 2022

Garsiya Evgeniy

Hai Le

28 October | 2022

Aims

To study the fMRI classification task and compare it to results obtained on sMRI. Provide experiments to improve accuracy metric. Study data preprocessing and explore its effect on classification results.

Pipeline

01. Data review & Preparation
02. Network architecture
03. Experiments
04. Results
05. Problems discussion
06. Links

Data review & Preparation (1): SRPBS Multidisorder Dataset

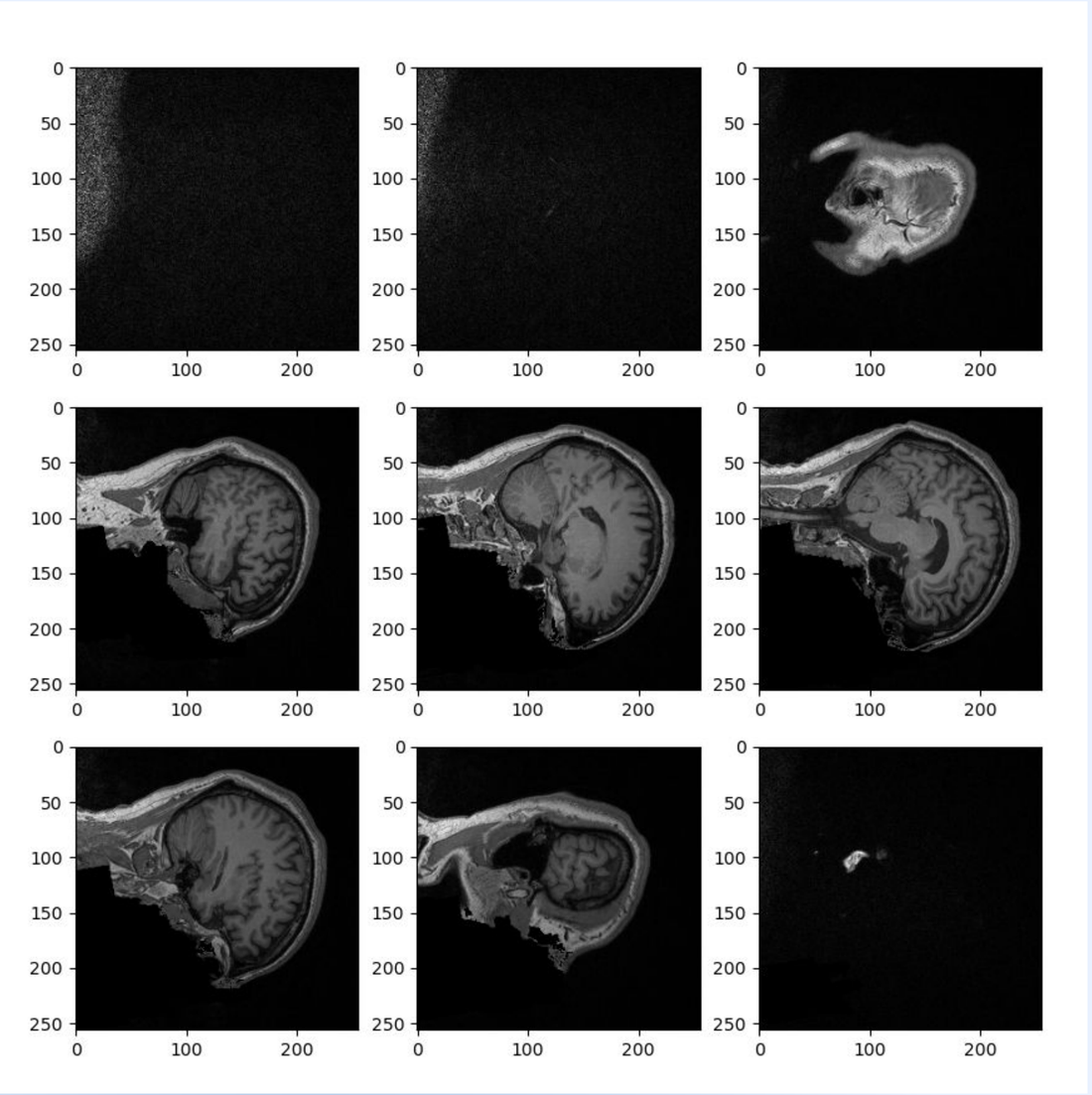
The 3T MRI imaging data from 1627 participants collected at 12 sites. Contains fMRI, sMRI and field map data.

```
SRPBS_1600
├── data
│    ├── sub-*
│    │    ├── rsfmri
│    │    ├── t1
│    │    └── fmap
│    ...
├── README.txt
├── participants.tsv
├── sup1.tsv
├── sup2.tsv
├── sup3.tsv
├── sup4.tsv
├── sup5.tsv
├── sup6.tsv
├── sup7.tsv
├── sup8.tsv
├── sup9.tsv
├── MRI_protocols_rsMRI.tsv
├── MRI_protocols_T1w.tsv
├── group_bold.tsv
├── group_T1w.tsv
└── deface_QC.tsv
```

All diagnosis:

- 0: Healthy Control
- 1: Autistic Spectrum Disorders
- 2: Major depressive disorder
- 3: Obsessive Compulsive Disorder
- 4: Schizophrenia
- 5: Pain
- 6: Stroke
- 7: Bipolar disorder
- 8: Dysthymia
- 99: Others

Data review & Preparation (2): sMRI data

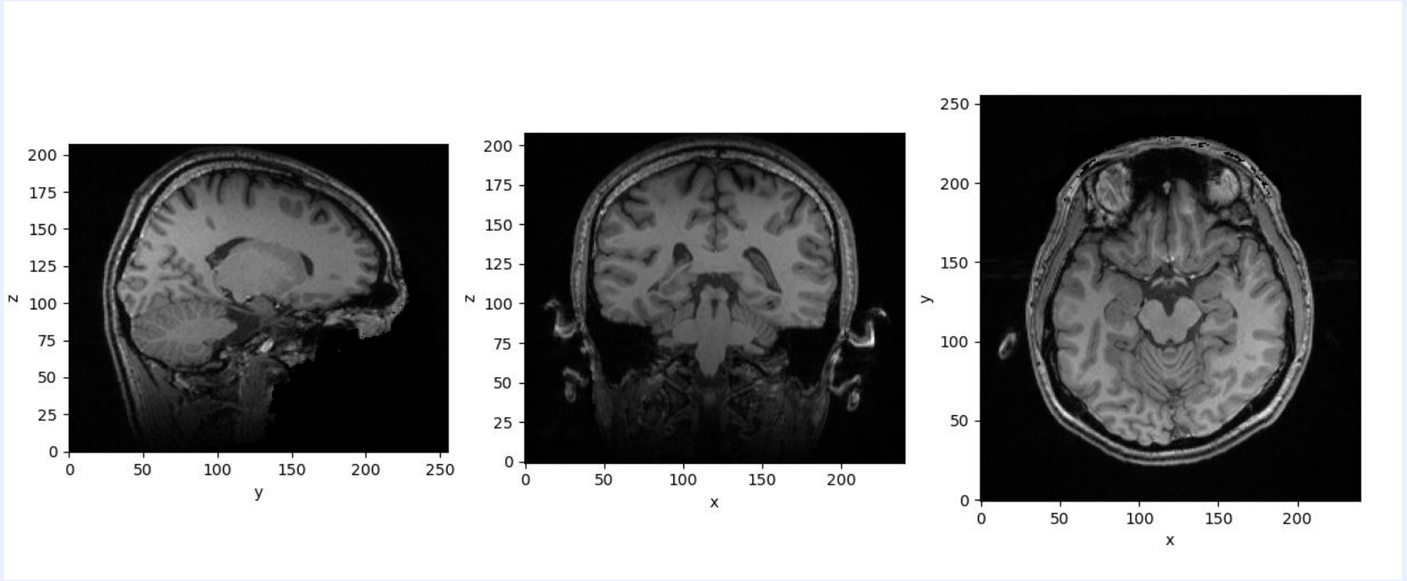


slices of one projection

Data shape: (240, 256, 256)

240 slices with 256 x 256 dimension

NIFTI data format



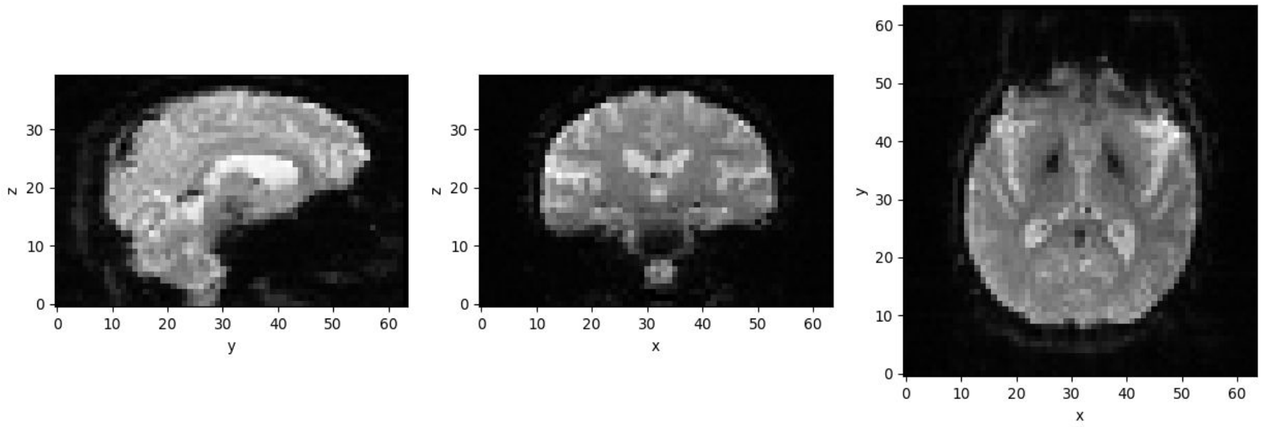
one slice of three projections

Data review & Preparation (3): fMRI data

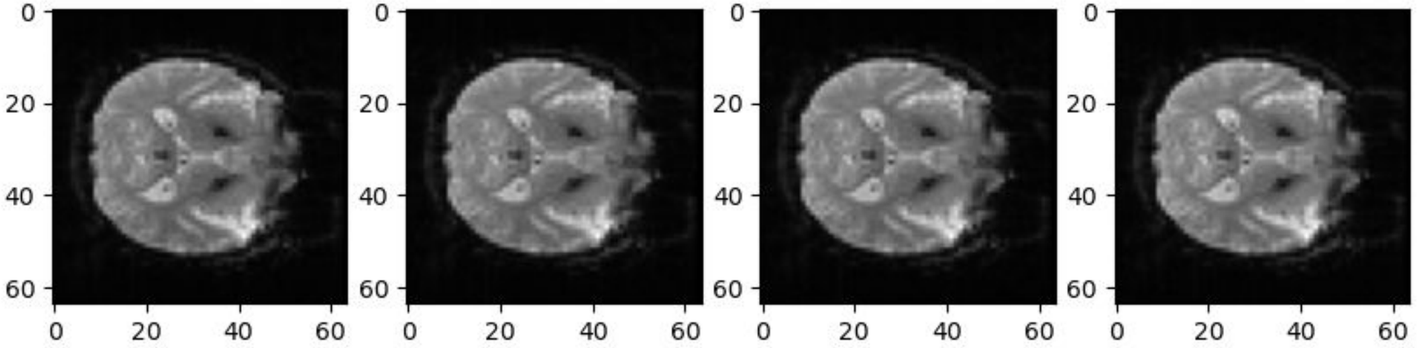
Data shape: (64, 64, 40, 240)

40 slices with 64 x 64 dimension during

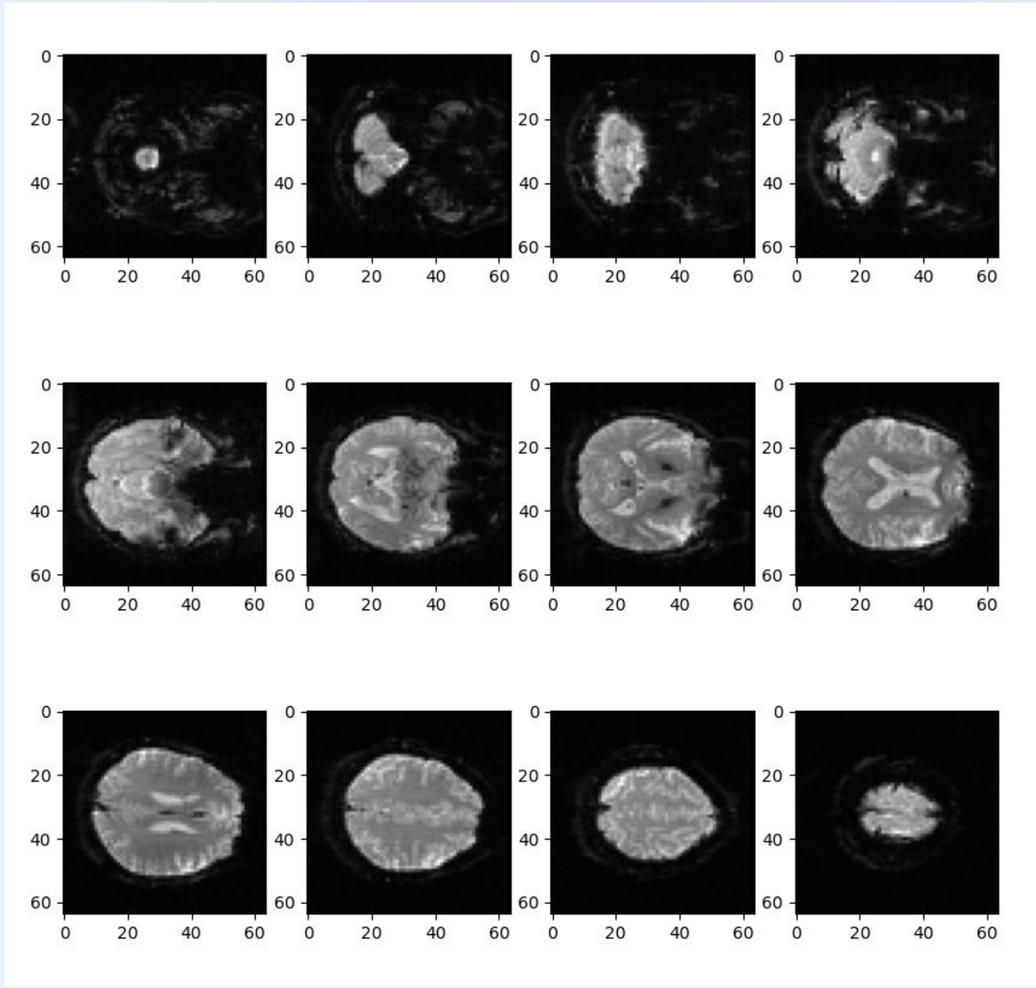
240 time steps



one slice for three projections



one projection slice through time points



slices for one projection

Data review & Preparation (3): Preprocessing Steps

sMRI

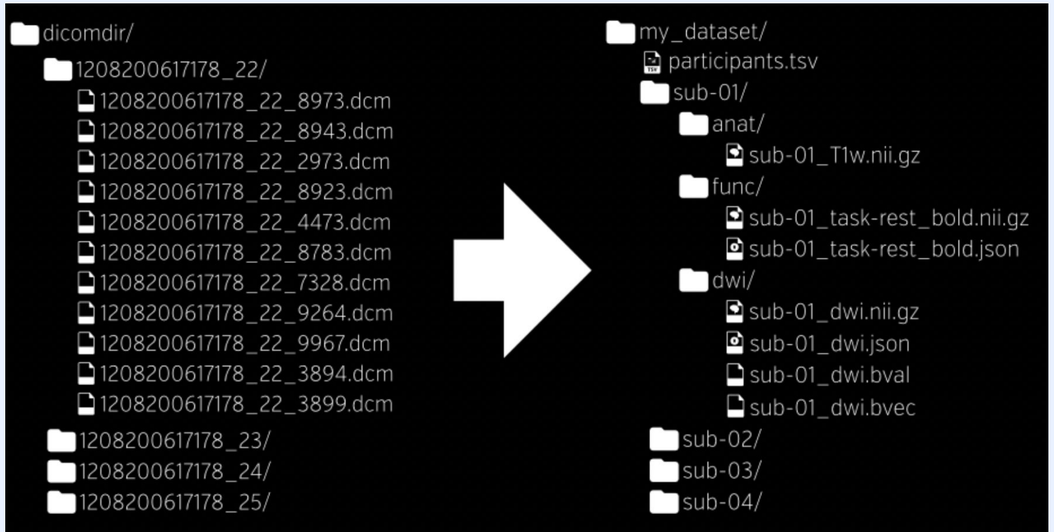
- bias fields correction
- t1 linear registration

fMRI

- concatenation of separated fMRI files to united raw data file for each patient
- fmriprep pipeline on raw data in BIDS format
- time series extraction with nideconv and nilearn on raw + fmriprep processed data
- time series concatenation

```
(base) evgeniygarsiya@MacBook-Pro-Evgenij-2 fmriprepfs_pipeline % docker build -t fmriprep_pipeline .
[docker run --platform='linux/amd64' --rm --name fmri --cpuset-cpus=0-3 -d  -v /Users/evgeniygarsiya/Desktop/NeuroML_proj/Project_d
ataset:/input -v /Users/evgeniygarsiya/Desktop/NeuroML_proj/Project_dataset:/output fmriprep_pipeline
[+] Building 1.2s (21/21) FINISHED
 => [internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 134B                                                                                    0.0s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                         0.0s
 => [internal] load metadata for docker.io/nipreps/fmriprep:20.0.1                                                      1.0s
 => [internal] load build context                                                                                       0.0s
 => => transferring context: 1.62kB                                                                                     0.0s
 => [ 1/16] FROM docker.io/nipreps/fmriprep:20.0.1@sha256:0bfaef1775318c92f9d70bd9b318455019de597561f2dae2a6149f67880617a0   0.0s
 => CACHED [ 2/16] RUN apt-get update -qq &&  apt-get install --yes --no-install-recommends  zip unzip pigz jq moreutils ti  0.0s
 => CACHED [ 3/16] RUN conda install -y -q -f -c conda-forge dcm2niix=1.0.20190902 &&  sync                             0.0s
 => CACHED [ 4/16] RUN commit=c9cf5ad16fb3021bb5c6e94ec5c63aec7ea9a99c &&  curl -LOk https://github.com/pieper/dicomsort/ar  0.0s
 => CACHED [ 5/16] COPY ./requirements.txt /code/requirements.txt                                                       0.0s
 => CACHED [ 6/16] RUN pip install --upgrade pip &&  pip install -r /code/requirements.txt                              0.0s
 => CACHED [ 7/16] RUN apt install --yes parallel                                                                       0.0s
 => CACHED [ 8/16] COPY freesurfer_license.txt /opt/freesurfer/license.txt                                              0.0s
 => CACHED [ 9/16] COPY entrypoint.sh avg_subject.zip /code/                                                            0.0s
 => CACHED [10/16] COPY heuristic.py /code/heuristic.py                                                                 0.0s
 => CACHED [11/16] COPY generate_norm.py /code/                                                                         0.0s
 => CACHED [12/16] COPY series_reg.py /code/                                                                            0.0s
 => CACHED [13/16] COPY watcher.sh /code/                                                                               0.0s
 => CACHED [14/16] COPY study_uid.py /code/                                                                             0.0s
 => CACHED [15/16] COPY expert.opts /code/                                                                              0.0s
 => CACHED [16/16] RUN chmod +x /code/entrypoint.sh && chmod +x /code/watcher.sh                                        0.0s
 => exporting to image                                                                                                  0.0s
 => => exporting layers                                                                                                 0.0s
 => => writing image sha256:46ad89801b0531f8ac4b0ba00129a68f1b01d59b23a17cf5c63e7e67da4de3a3                            0.0s
 => => naming to docker.io/library/fmriprep_pipeline                                                                    0.0s
7c05eccf2334d2e6cd1bfa43da173fda15a8a5ec9a86058b8c89d74735be8b6a
(base) evgeniygarsiya@MacBook-Pro-Evgenij-2 fmriprepfs_pipeline % docker ps -a                                         ]
CONTAINER ID   IMAGE               COMMAND                  CREATED         STATUS                     PORTS     NAMES
7c05eccf2334   fmriprep_pipeline   "bash -c 'source /co…"   5 seconds ago   Up 3 seconds                         fmri
7839e304b622   pred_pain_app       "flask run --host 0.…"   6 days ago      Exited (0) 6 days ago                pred_pain_app_evgeni
```

docker run example



BIDS format

Network Architecture

From seminar 6

RCNN based 3D classification

For best architecture:

Augmentations: (next slide)

Optimiser: Adam (lr=1e-4)

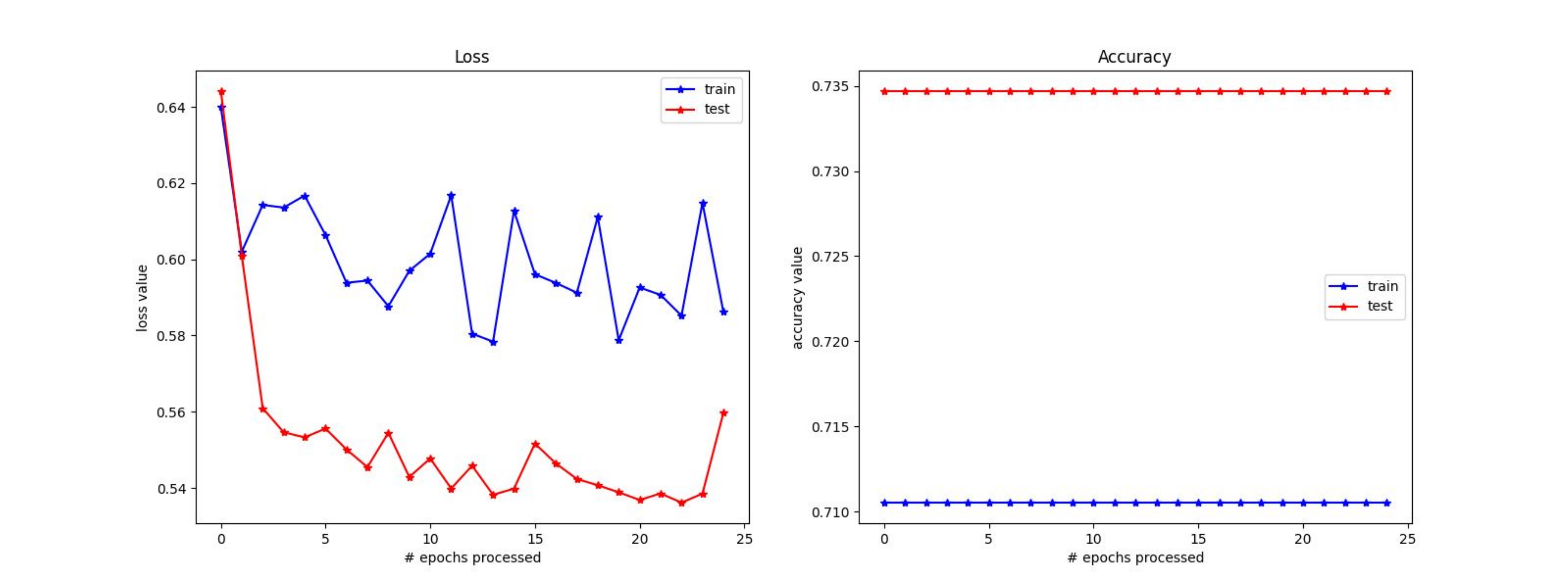
Scheduler:

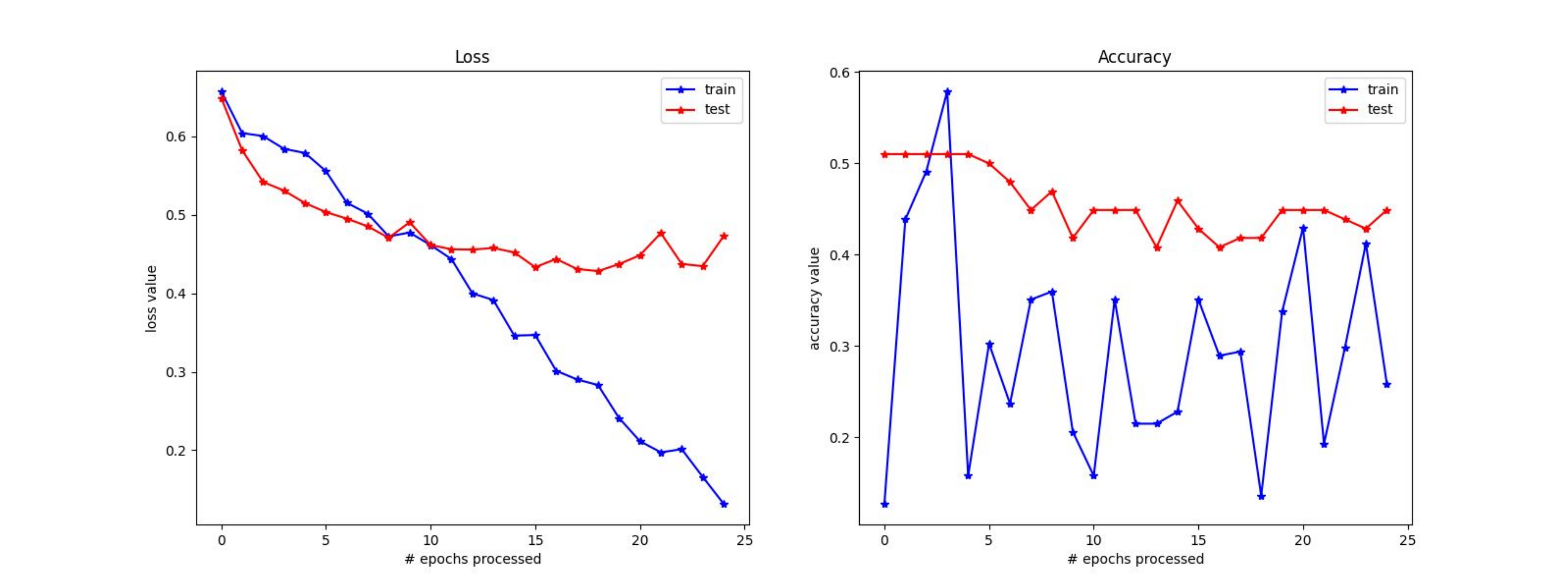
CosineAnnearlingLR(T_max=20)

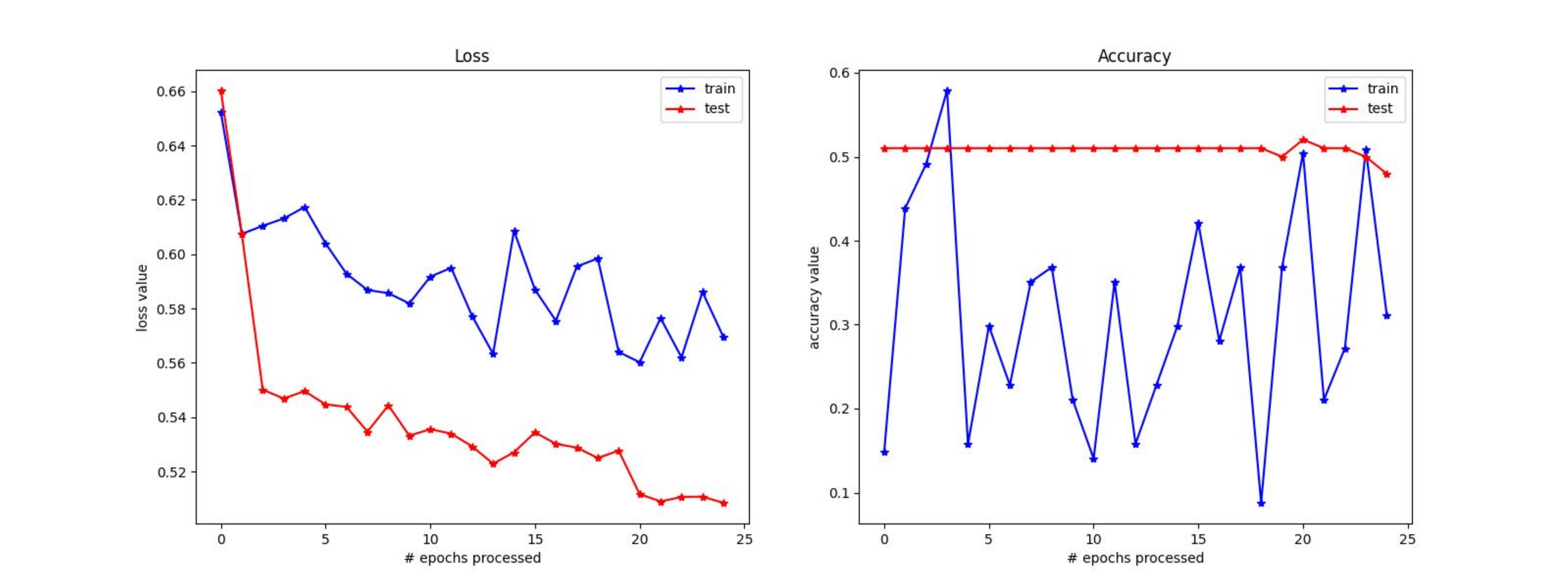
```
class FMRINET(nn.Module):
    def __init__(self, in_channels=10,
                 out_channels_list=[32, 64, 128, 256],
                 network_type ='Conv',
                 act ="relu",
                 norm= "batch",
                 input_shape=(64, 64, 64),
                 n_outputs =2,
                 n_fc_units=128,
                 hidden_size=128,
                 dropout =0.2,
                 n_layers=1,
                 is_rcnn=False):
        super(FMRINET, self).__init__()
        self.is_rcnn=is_rcnn
        if self.is_rcnn:
            self.cnn =ConvEncoder(in_channels=1,
                                  out_channels_list=out_channels_list,
                                  network_type =network_type,
                                  act =act,
                                  norm= norm,
                                  input_shape=input_shape,
                                  is_rcnn=True)
            self.gru =ClfGRU(self.cnn.n_flatten_units,
                             in_channels,hidden_size=hidden_size,n_layers=n_layers)
            self.fc =nn.Sequential(
                              nn.Dropout(dropout),
                              nn.Linear(self.gru.gru_out_size, n_fc_units),
                              nn.ReLU(inplace=True),
                              nn.Linear(n_fc_units, n_outputs))

        else:
            self.cnn =ConvEncoder(in_channels=in_channels,
                                  out_channels_list=out_channels_list,
                                  network_type =network_type,
                                  act =act,
                                  norm= norm,
                                  input_shape=input_shape,
                                  is_rcnn=False)
            self.fc =nn.Sequential(
                             #  nn.Dropout(dropout),
                              nn.Linear(self.cnn.n_flatten_units, n_fc_units),
                              nn.ReLU(inplace=True),
                              nn.Linear(n_fc_units, n_outputs))

    def forward(self,x):
        if self.is_rcnn:
            x =self.cnn(x)
            x =self.gru(x)
            x =self.fc(x)
        else:
            x =self.cnn(x)
            x =self.fc(x)
        return x
```

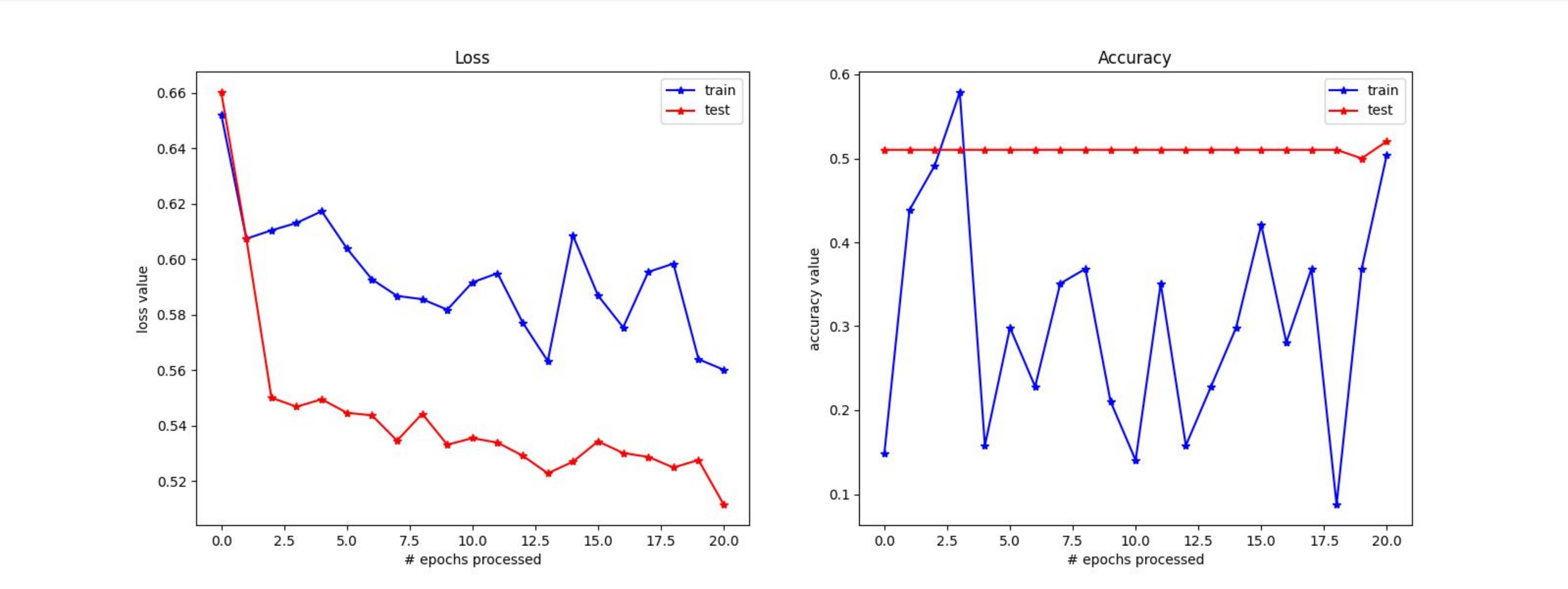
Experiments Strategy (1)

While developing and training our model initially, we stumbled across a <u>strange problem -></u> experimentation with different stack of augumentation









Experiments Strategy (2)

Once we figured out a suitable stack of transform that negate the mentioned issue, we kept the same augmentations. We used **MONAI transformations**.

Our augmentation looked like the following:

```
train_transforms = Compose(
    [
        LoadImaged(keys="image"),
        EnsureChannelFirstd(keys="image"),
        CropForegroundd(
            keys="image", source_key="image", select_fn=lambda x: x > 0, margin=0
        ),
        NormalizeIntensityd(
            keys="image",
        ),
        RandAffined(keys="image"),
        Orientationd(keys=["image"], axcodes="RAS"),
        RandBiasFieldd(keys=["image"]),
        RandGaussianSmoothd(keys="image"),
        RandFlipd(keys="image"),
        Resized(keys="image", spatial_size=(32, 32, 32)),
    ]
)
```

```
val_transforms = Compose(
    [
        LoadImaged(keys="image"),
        EnsureChannelFirstd(keys="image"),
        CropForegroundd(
            keys="image", source_key="image", select_fn=lambda x: x > 0, margin=0
        ),
        NormalizeIntensityd(
            keys="image",
        ),
        Resized(keys="image", spatial_size=(32, 32, 32)),
    ]
)
```

MONAI

SCAN ME

Experiments Strategy (2)

With the stack of augmentations from the prev. slide, our experimentation strategy were as follow:

Raw sMRI

- No learning rate scheduler
- With learning rate scheduler
- ~20 epoches

Raw fMRI

- No learning rate scheduler
- With learning rate scheduler
- ~20 epoches

Processed sMRI

- No learning rate scheduler
- With learning rate scheduler
- ~20 epoches

Processed fMRI

- No learning rate scheduler
- With learning rate scheduler
- ~20 epoches

Experiments Raw sMRI Results

N	Name	Description	Best Accuracy
1	Initial sMRI	25 epoches, bad augs part 1 (static train & val acc), no scheduler, Adam optimiser (default params)	~51.02%
1	Initial sMRI	25 epochs, bad augs part 2 (static train & val acc), no scheduler, Adam optimiser (LR=1e-3)	~73.05%
3	Opt. Augs sMRI	optimized augs, no scheduler, optimized Adam optimiser (LR=1e-4)	~71.5%
4	Opt. Augs & Scheduler sMRI	15 epoches, optimized augs, CosineAnnearlingLR scheduler, optimized Adam optimiser (LR=1e-4)	~77.55%

Experiments Preprocessed sMRI Results

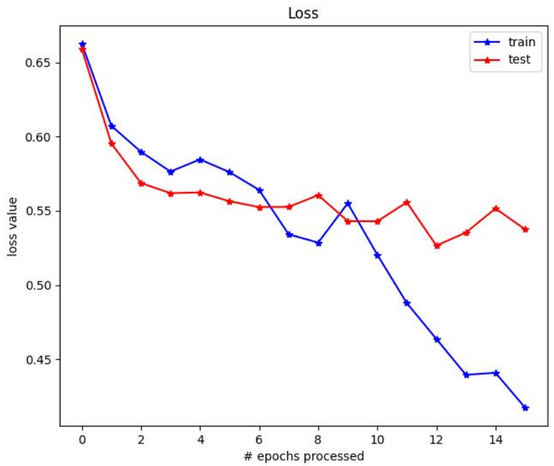
N	Name	Description	Best Accuracy
1	Preprocessed sMRI & No Scheduler	optimized augs, no scheduler, optimized Adam optimiser	~71.43%
2	Preprocessed sMRI & Scheduler	(17 epoches) optimized augs, with scheduler, optimized Adam optimiser	~77.55%

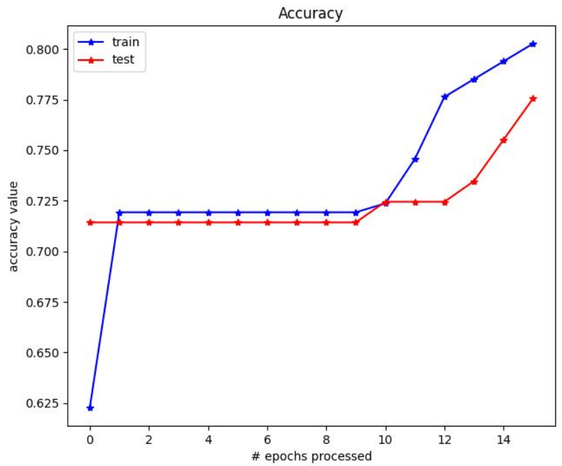
Experiments Raw fMRI Results

N	Name	Description	Best Accuracy
1	Raw fMRI & No Scheduler	9 epoches, raw data, no augs, no scheduler, Adam optimiser (default params)	~70.02%
2	Raw fMRI & Scheduler	9 epoches, raw data, no augs, no scheduler, Adam optimiser (default params)	~71.43%

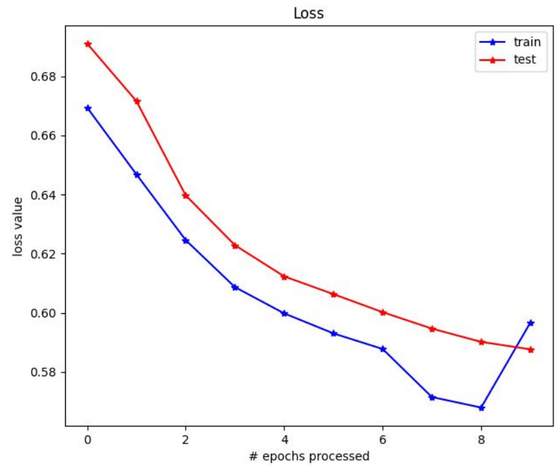
Results & Comparison

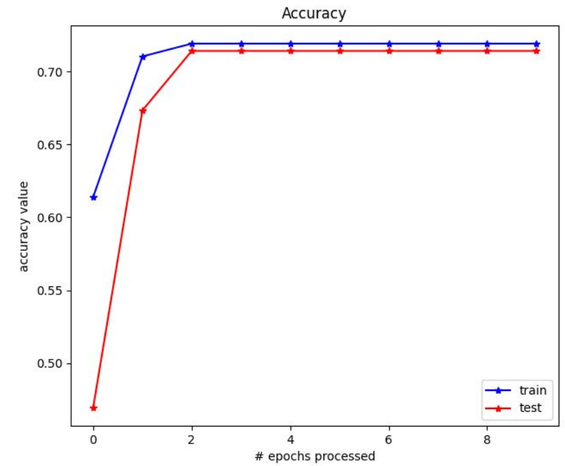
sMRI





fMRI



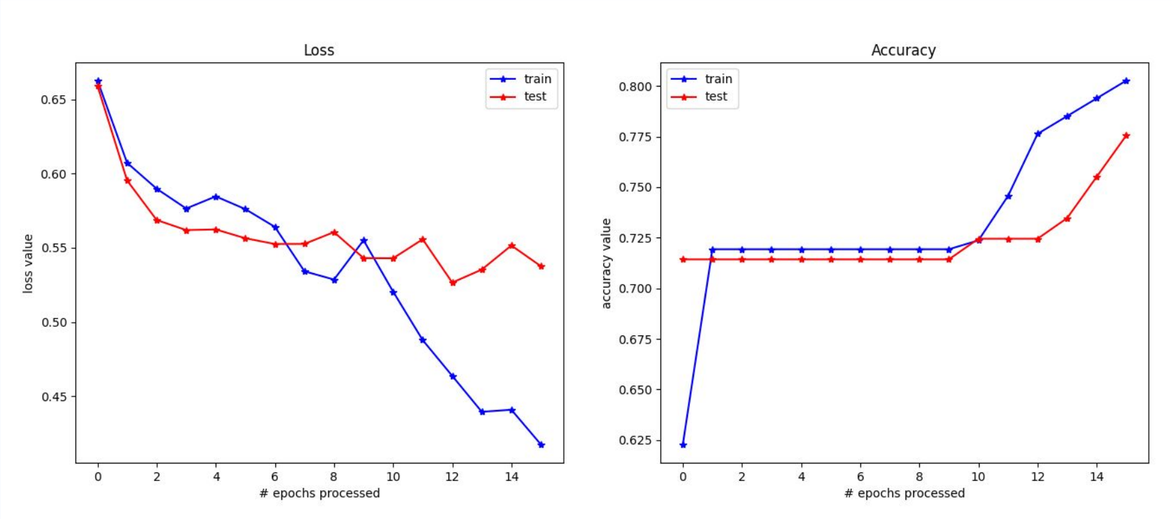


- sMRI raw data performs in quite a better way comparing to raw fMRI considering similar augmentations and parameters.

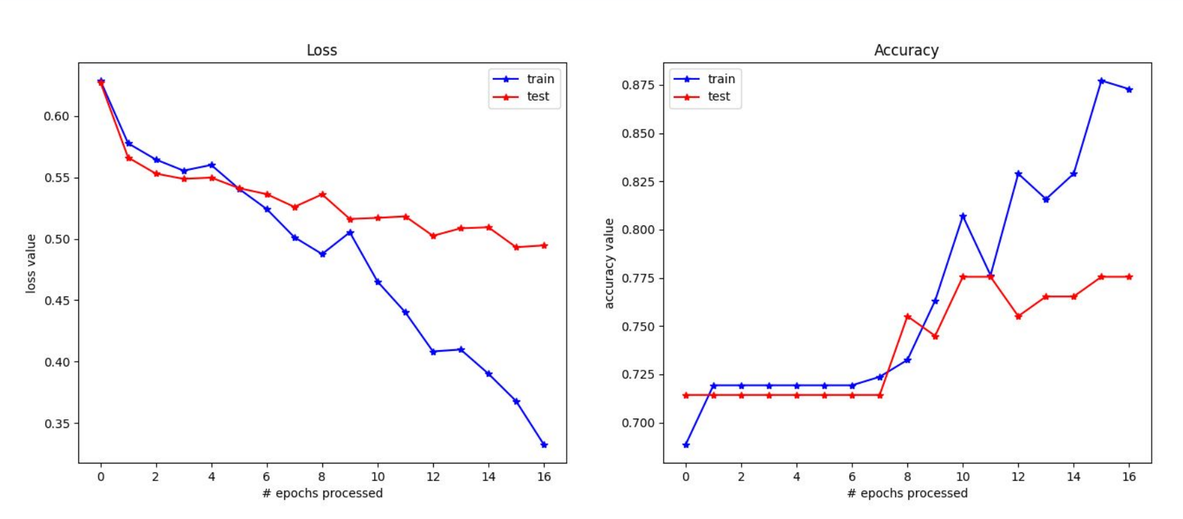
Results & Comparison

- The best results were obtained on raw and processed sMRI with optimised augs and scheduler. The best values are similar but behaviour differs.

raw



processed



- Raw data in this case behaves better. The possible reason is the fact that parameters for raw data could be tuned much better then for processed one. But in general we expect processed data to perform much better. And it is possible if we take a look on a train metric

Results & Comparison

- The worst was obtained on the raw sMRI without optimised augs and any tuning.
- Comparing sMRI and fMRI raw data on tuned parameters they performs quite similar in terms of best metric, but behaves different: fMRI accuracy is too static.
- The sMRI preprocessing didn't result in any improvement in our case but changed the metric behaviour.
- Adding scheduler results in quite a good improvement.
- The most important improvement obtained on optimised augs + sheduler
- Possible further experiments:

  - scheduler and optimiser parameters tuning
  - batch size and sampler tuning
  - different preprocessing
  - adding loss weights to deal with imbalance

Discussion

Problems we faced:

- Big amount of data forced us to reduce it by choosing only 300 samples from two sites.
- The task difficulty made us to solve a binary classification task, while dataset provides a multi labels classification.
- Quite a time consuming training process (even on a server gpu) made it difficult to provide a lot of experiments
- Data preprocessing is too effects the MRI classification
- MRI data and structure is quite complicated
- Provided dataset was not in the BIDS format which means it took time to reformat the structure and make a datasets suitable for running fmripreproc framework on it
- Time consuming preprocessing pipeline
- Unfortunately due to time consuming fMRI processing we didn't succeed in getting a preprocessed fMRI data and launching experiments on it

Discussion

Problems we solved:

- We studied real sMRI and fMRI data and explored how to work with it
- We became familiar with 3D classification task and studied the way to work with 3D networks.
- We studied the image formats used in MRI and dataset structures required for preprocessing frameworks.
- We launched and made work the fmripreproc framework through docker on fMRI data.
- We provided several experiments, solved several bugs and errors during training process and obtained an accuracy improvement comparing to initial raw data launch on default parameters

Conclusion

1. Our aim was to explore what's the highest data quality can be obtained from raw data with all sorts of approaches, considering the lack of processing time or tools
2. The project helped us to deal with data types and its forms that are required for the network.
3. Comparing to homeworks and course in general, in practice it turned out that MRI image processing and training pipeline is much more difficult and incomprehensible.
4. We figured out the preprocessing, launched it, but since this is a rather long process, we did not have time to try to train the net.

Links

Github:

https://github.com/highly0/SRPS160_Neuro_ML

Included our code, README on how to recreate the results, & plots