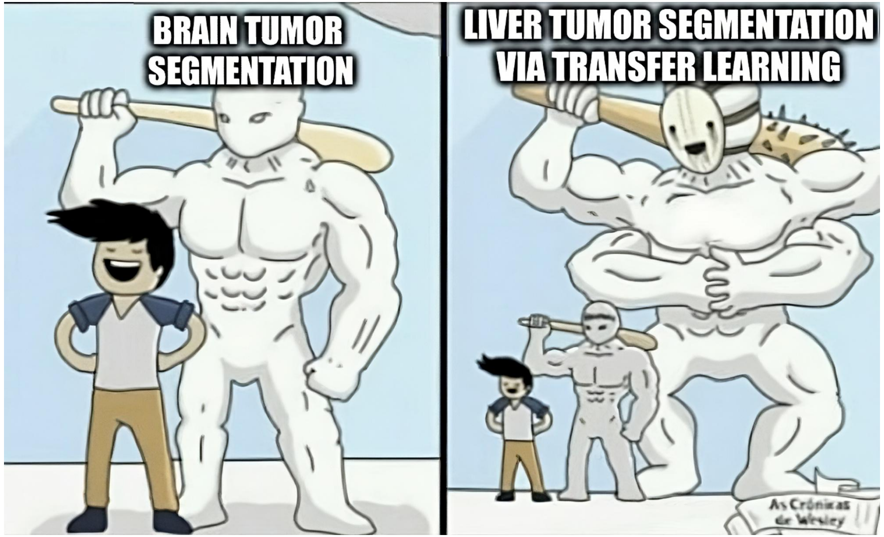
Braver Cat

Brain and Liver cancer segmentation via Transfer Learning

Daniele Solombrino, Emanuele Volanti

Project idea

- Basic: **brain** tumor segmentation
  - **End-to-end** train
- Advanced: **liver** tumor segmentation
  - **Transfer Learning**
    using NN trained in brain step



Motivations

- Work on a typical Computer Vision task
  - **Segmentation**
- Experience a full **Deep Learning** pipeline
  - Data gathering
  - Data pre-processing
  - Model engineering
  - Model evaluation
  - Result analysis
- Very **small** datasets for very **specific** pathologies

Paper hunt

- Literature research process
  - Find interesting papers in **surveys**
  - **Deep dive** on specific papers
- Literature review results
  - "Cheplygina et al., 2019" → shallow models 👎
  - "A Survey on DL in Medical Image Analysis" → Deep Learning □□
  - "GANs for Medical Image Analysis" → computationally expensive 🤌
  - "Adversarial Methods in Medical Analysis" → computationally expensive 🤌
  - "**Brain Tumor Segmentation with DNNs**" → DL 👍, computationally feasible 👍, novelties👍

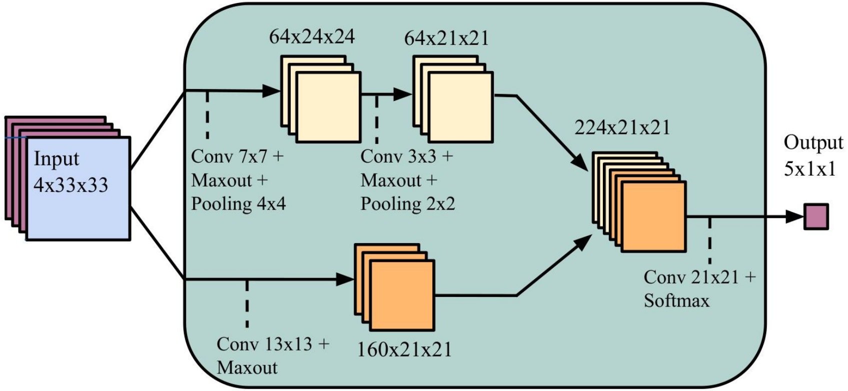
Why “Brain Tumor Segmentation with DNNs”

Paper introduces interesting and non-trivial **novelties**:

1. TwoPathCNN
2. CNN concatenation
3. Patched inputs
4. Dual-stage training as unbalanced data remedy
5. Maxout activation function → arXiv:1302.4389
6. Convolutional output layer

TwoPathCNN

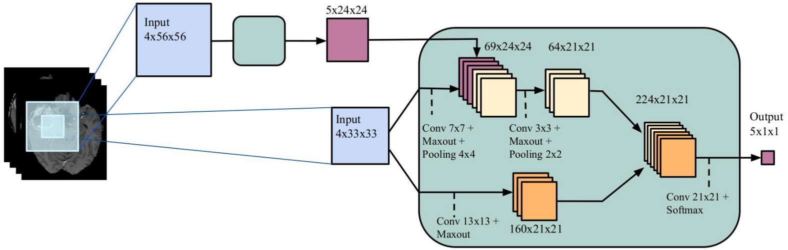
- Multiple filter scales adoption → **different scale** representations
- Combining them → CNN learns much **more** feature **interactions**
- Medical domain: non-local, potentially unknown **tissue relationships**
- Filter and **path-level*** computational **parallelism**

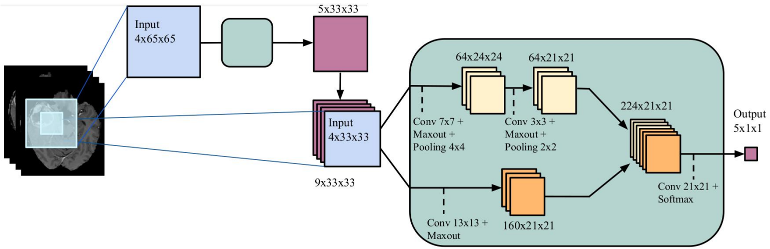


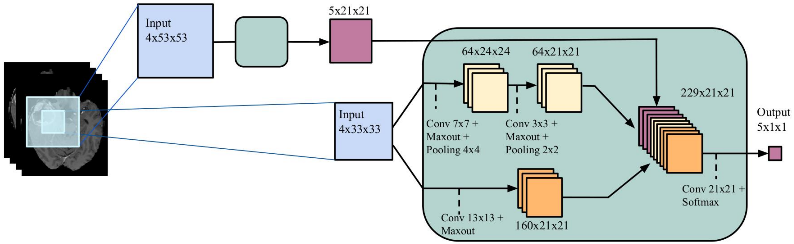
* if supported by Deep Learning framework

CNN concatenation

- TwoPathCNN: multi-scale interactions, single input…
- … what about **multiple inputs**?
- Boosts TwoPathCNN benefits
- Multiple concatenation strategies
    - Task-dictated

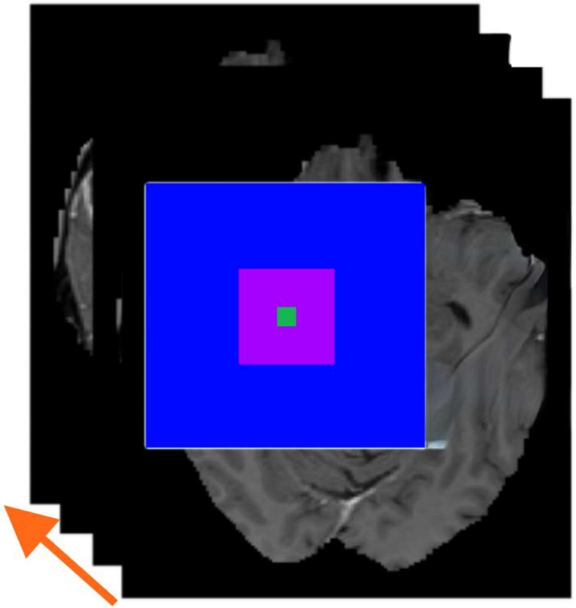






Dataset and patched inputs

- Dataset → brain scans w/ segmentation masks
- **4-channeled** inputs → 4 CT modalities
- 5 labels → possible tumor stages
- For every **pixel p**, model:
  - Sees **differently**-**scaled** patches centered on p
  - Predicts segmentation label for pixel p

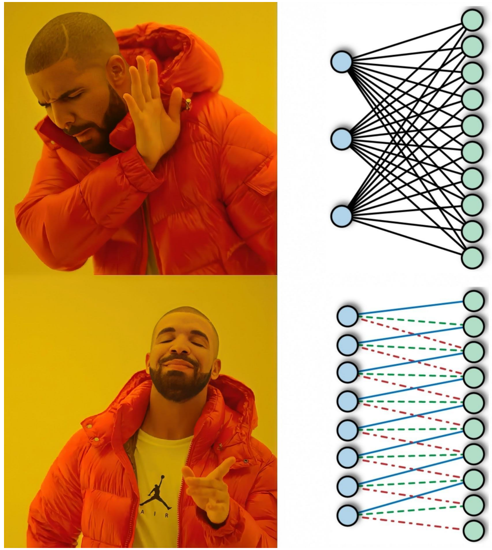


Dual-stage training as unbalanced data remedy

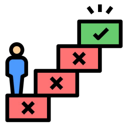
- **Unbalancement** towards “non-tumor” label 👎
- 💡 Idea: **dual-stage training**
- 1st stage → balanced data
- 2nd stage → unbalanced data
  - Resume from 1st stage w/ all but last layer frozen

Convolutional output layer

- Typical CNNs use a **fully-connected** output layer
  - Needs 1D **flattening** → wastes time and memory 👎🏿
  - Can **not** be **parallelized** 👎🏿
- 💡 Idea: 1x1 **conv** layer, w/ c channels
  - c → number of classes
  - Way easier to **parallelize** 👍🏿
  - Weight sharing 👍🏿

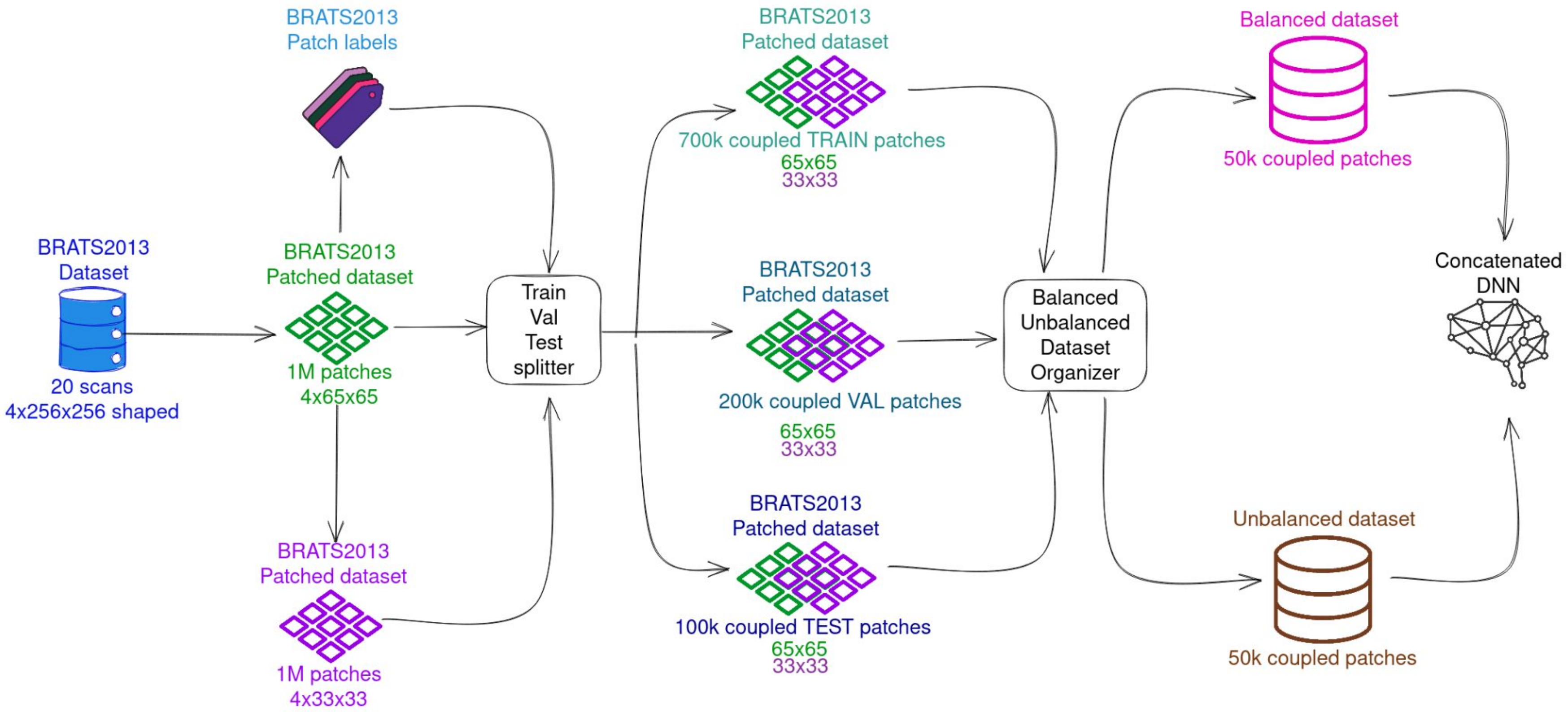


Challenges



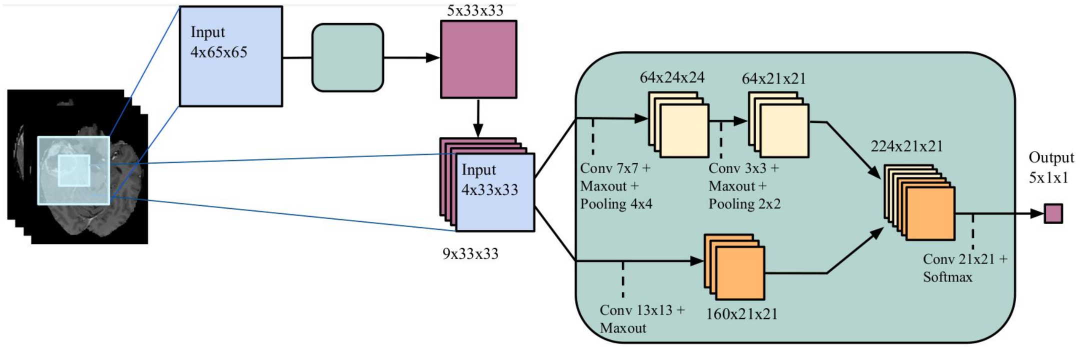
- Complex pre-processing
- Custom Neural Networks from scratch
- Hyperparameter tuning
- Transfer Learning
    - What layers to freeze?
    - Input channels mismatch

Non-trivial pre-processing

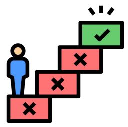


Custom Neural Networks from scratch

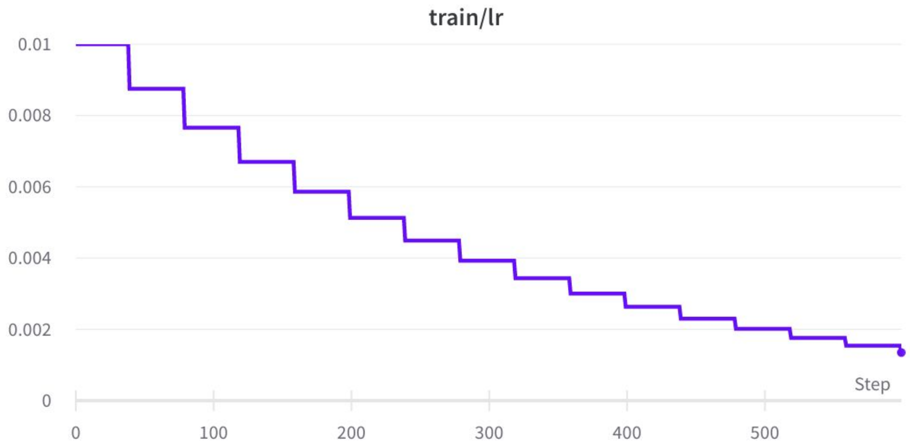
- Defined **PyTorch** modules for TwoPathCNN
- **Maxout** activation → **no** out of the box PyTorch **support**
- Implemented InputCascadeCNN model → two **concatenated** TwoPathCNN



Hyperparameter tuning

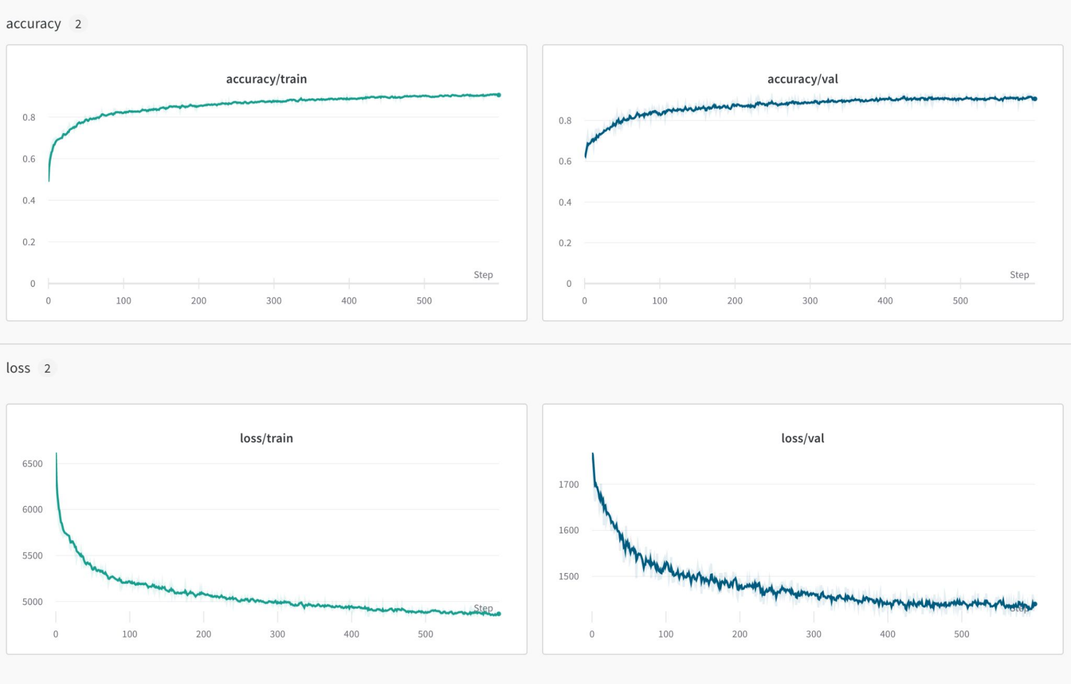


- Paper lists hyperparameters…
- … but
  - No values disclosed, or
  - Proposed values **perform badly**
- … so we worked on
  - Weights init method, dropout
  - Optimizer, LR w/ decay, momentum
  - Regularization
  - Input normalization, batch size

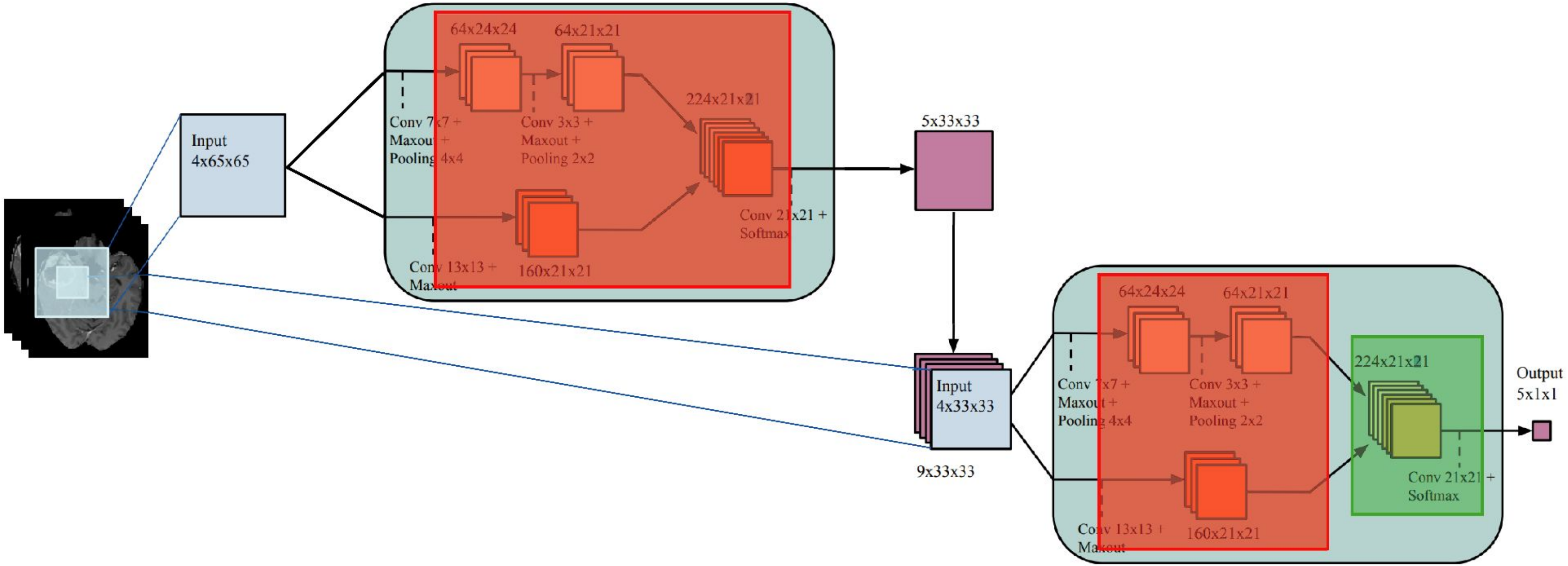


End-to-end dual-stage training

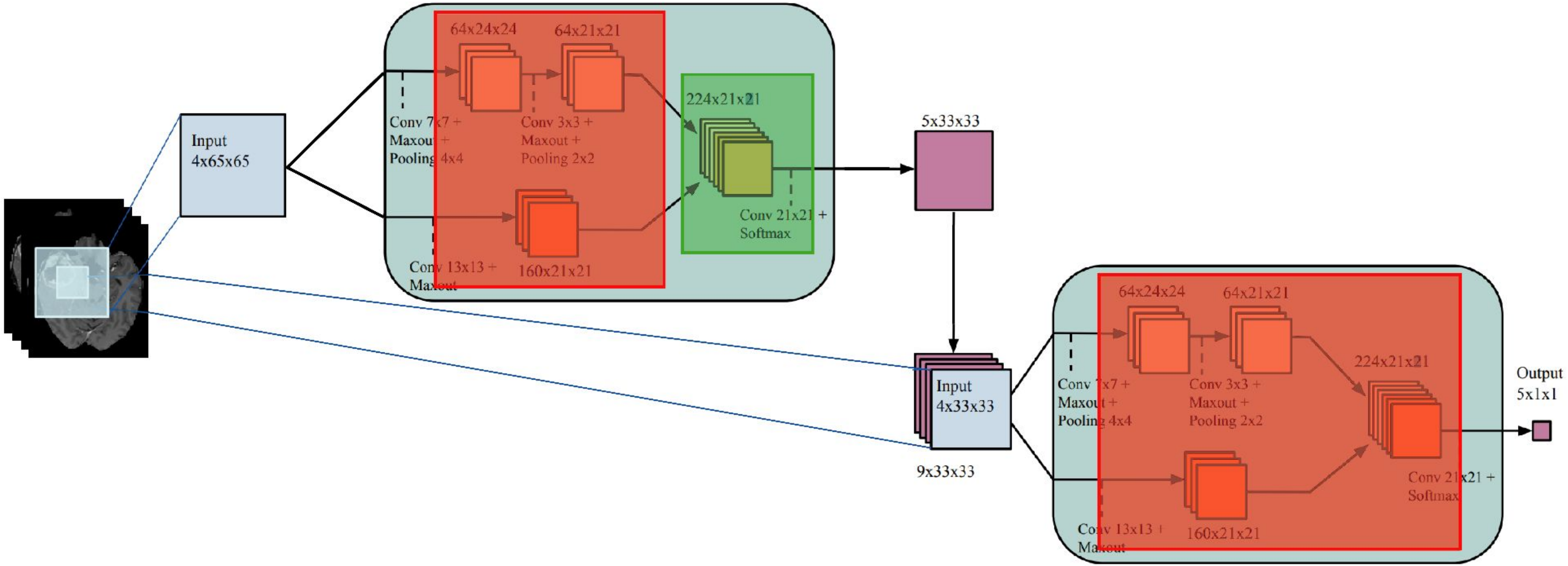
- 1st stage
  - ~2.2M trainable parameters
  - 600 epochs
  - ~5h, 30s per epoch
  - **>90%** train, val and test acc
- 2nd stage
  - ~600k trainable parameters
  - 300 epochs
  - 40 mins, 4s per epoch
  - **>92%** train, val and test acc



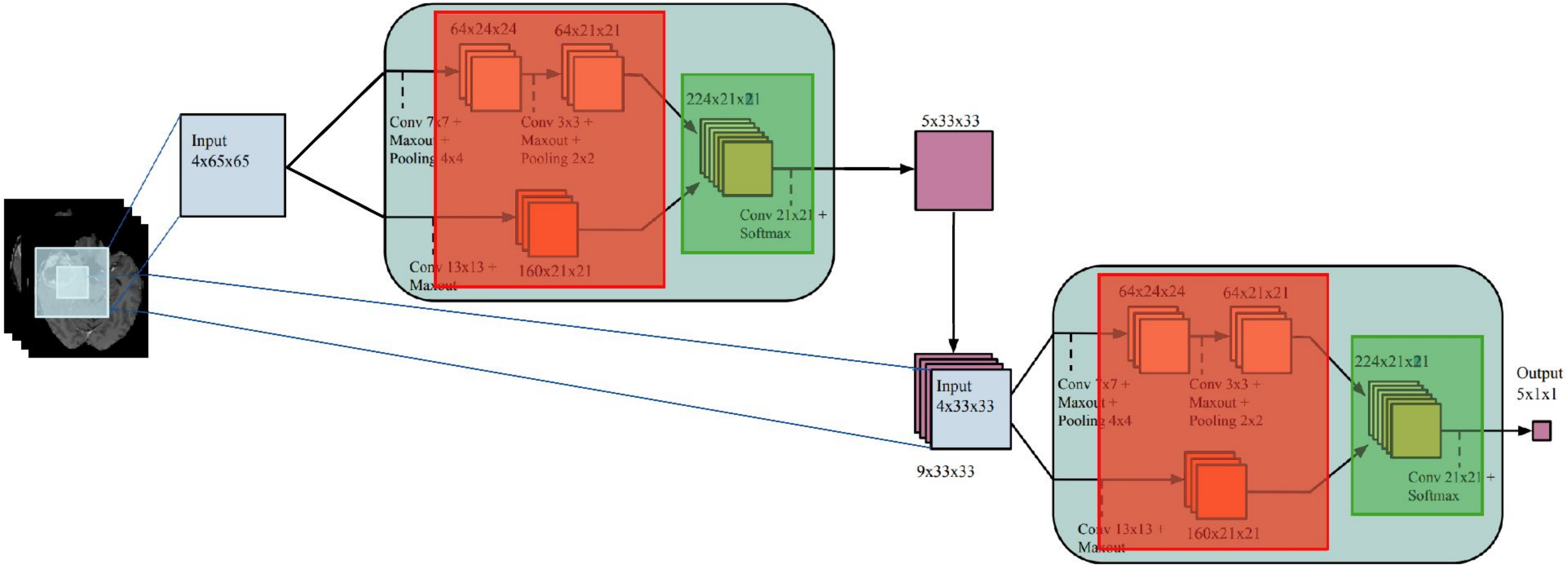
Transfer Learning: local scale



Transfer Learning: global scale



Transfer Learning: local and global scales



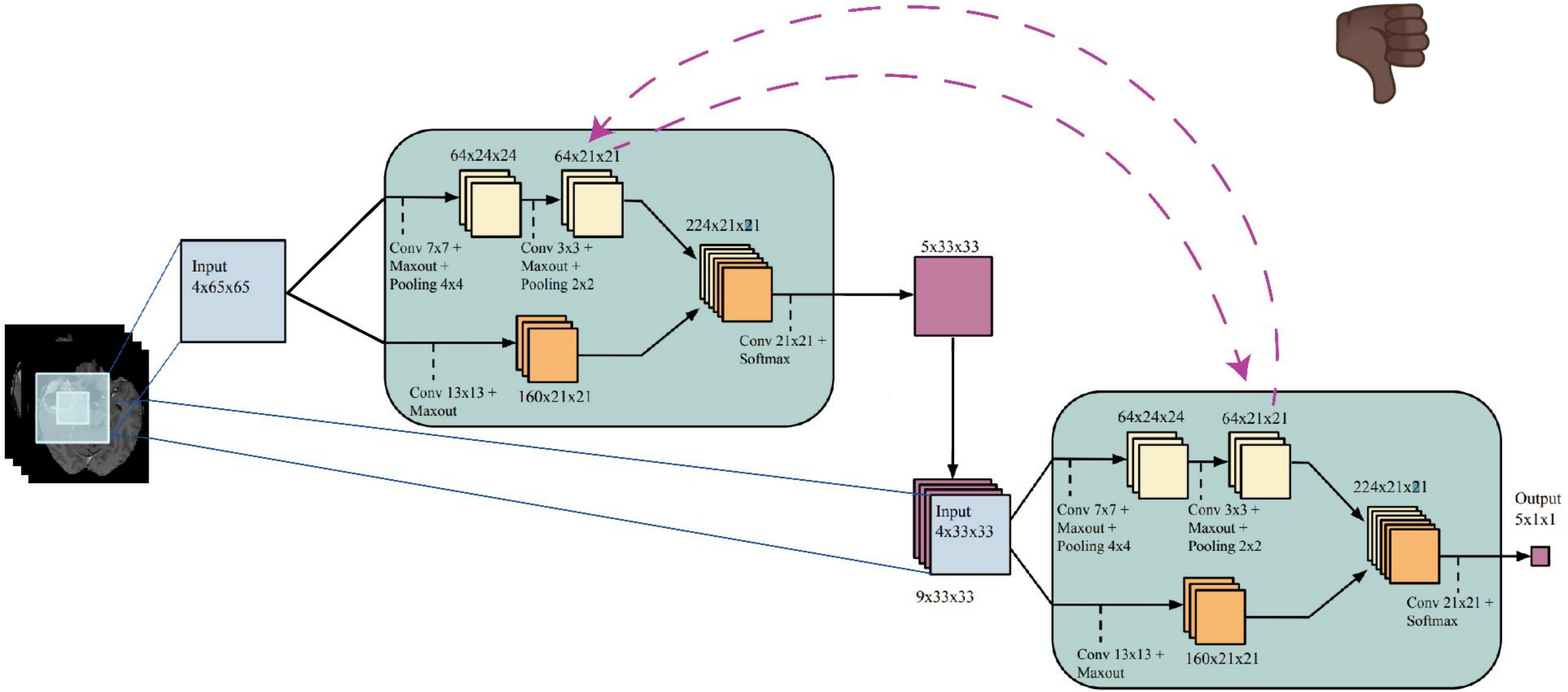
Transfer Learning: results

Transfer Learning setup	# trainable params	Epoch train time	Train accuracy	Validation accuracy
Local scale	600k	4s	52%	48%
Global scale	600k	4s	66%	61%
Global & Local scale	1.2M	9.5s	87%	**85%**
E2E	2.2M	30s	89%	86%

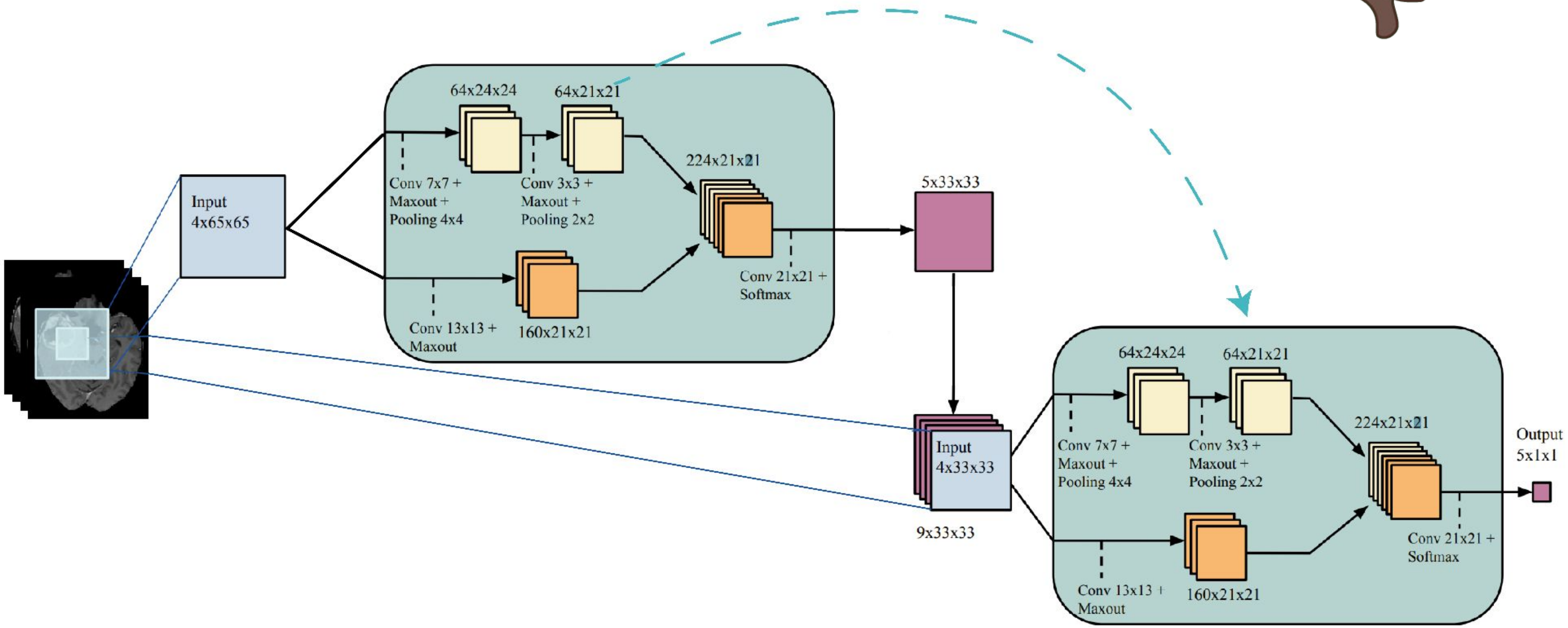
Parameter pruning?

- 💡 Idea: are **all** learnt **parameters** actually **useful**?
- Test time → replace some layers with **identity** operation
- Same TL configs → 👎🏿
- Keep local NN only → 👎🏿
- Keep global NN only → 👎🏿

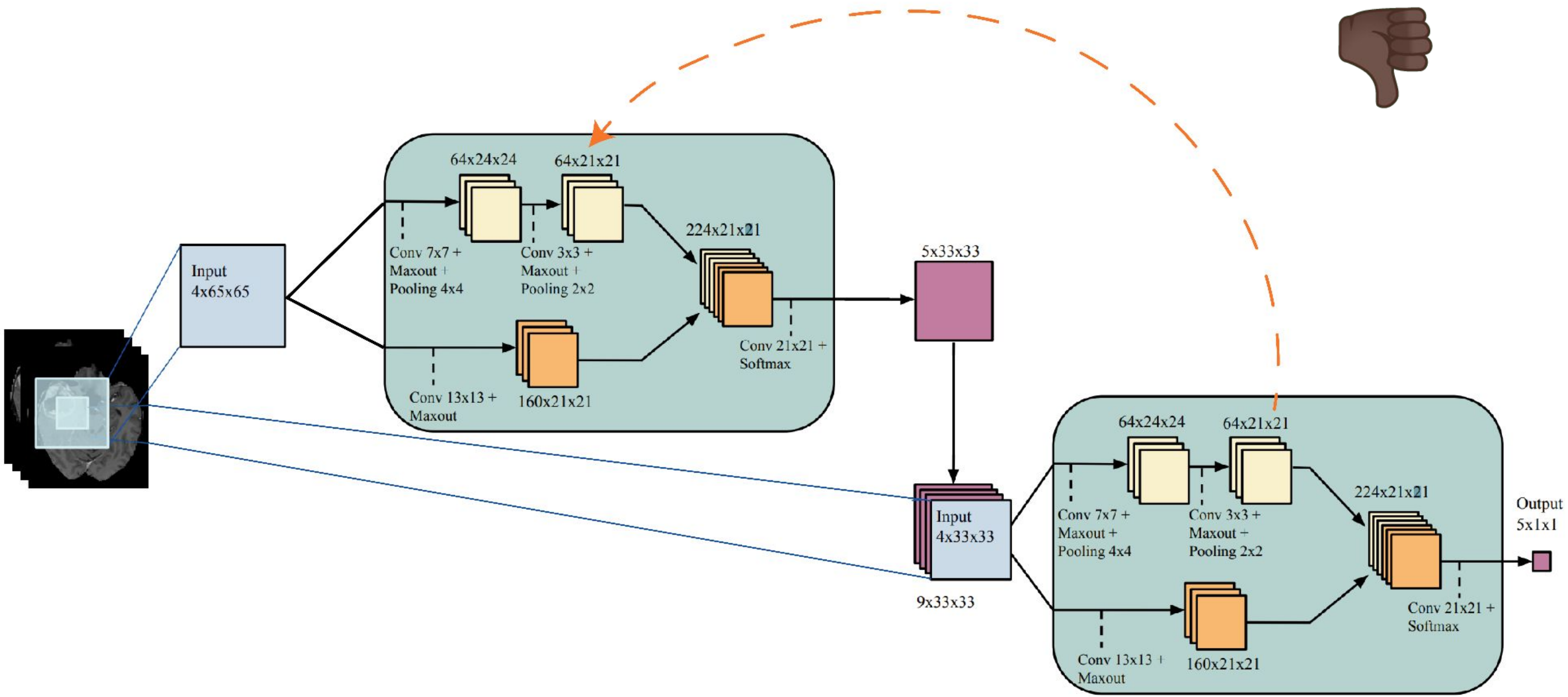
Layer mix-and-match: switch



Layer mix-and-match: copy



Layer mix-and-match: copy



Bonus section

- Data **exploration** → NiBabel
- Data **cleaning**
- Data pre-processing **parallelization**
- **Mean** and **std** computation
- **Weights and Biases** tracking 💜
- Custom-made training **dashboard** 💜

Recap

- Brain and liver tumor segmentation via Transfer Learning
- Paper **novelties**:
  - TwoPathCNN, CNN concatenation, maxout activation, conv out layer, dual-stage training
- E2E brain tumor segmentation
- Multiple TL setups for liver tumor segmentation
  - **1/3** training **time** → **-1%** train and validation accuracy w.r.t. E2E
- **Challenges**:
  - Pre-processing, custom modules, hyperparam tuning, Transfer Learning
- Brain NN at test time **not invariant** w.r.t.
  - Parameter pruning, layer “mix-and-match”