Brain Tumor Classification via Convolutional and Capsule networks

Satyarth Mishra Sharma, Artem Moskalev, Artur Grigorev

Skolkovo Insititute of Science and Technology, Russia

2018
March

1 Introduction

Tumor classification task is a long-standing problem in the field of medical images analysis. Determining the type of a brain tumor in its early stage is very important because it allows to adapt and adjust necessary treatment early and leads to a better prognosis.

Currently, there is a variety of techniques used. Many of them are based on Convolutional Neural Networks. However, the task of tumor classification is not claimed to be solved, and such problems as the lack of training data, imbalanced dataset, sensitivity to data transformations are still actual.

In this project, we investigate the abilities of recently presented Capsule Networks [2] to perform an accurate classification of brain tumors and compare them with Convolutional Neural network based approaches.

2 Data

2.1 Dataset

We use the same dataset as in [1], which consists contains 3064 T1-weighted contrast-enhanced images with three kinds of brain tumor: meningioma, glioma and pituitary tumor. The distribution of the classes is 24%, 46%, 30% respectively. Along with the images of the injured brain and its labels, we are also provided with tumor masks, which is basically a binary image that delineates a tumor region.

2.2 Preprocessing

As preprocessing, we employed histogram equalization with 2 bins and downsampling from $(512, 512)$ to $(64, 64)$. First, showed a great effect, enhancing an accuracy of classification. Second, the difference between prediction quality and dimensionality of input image $(256, 256)$ / $(128, 128)$ / $(64, 64)$ appeared to be negligible, so we chose the smallest image size to proceed because of computational reasons.

3 Compared models

3.1 Vanilla CNN

As in [1], we began with building simple Convolutional Neural network. Based on the approach from [3], we employed the following architecture (Non linearity is included between each convolution and maxpooling): Convolution $\rightarrow$ Maxpooling $\rightarrow$ Convolution $\rightarrow$ Maxpooling $\rightarrow$ 2 Fully connected dense layers with 800 ReLU neurons in each and then 3 output sigmoid neurons for each class. For convolutions we used $(5, 5)$ kernels, with no padding and unary strides, for Maxpooling kernel size was standard $(2, 2)$.

We trained Vanilla CNN on the parameter setting that proved best during experiments:

- 7 epochs
- batch size $= 25$
- ADAM optimizer with learning rate $= 1e - 5$

3.2 DenseNet

The second architecture to try was Densely Connected Convolutional Network (DenseNet) [2]. We came to this approach when tried to improve our Vanilla CNN with residual connections, which, in fact, gave no enhancement, except a bit reduced training time.

DenseNet consists of so-called dense blocks, separated by convolutional layers and maxpooling. Each dense block consists of dense units, which can be divided into two types: sequential units and bottleneck units. Sequential units use the pipeline of batch-normalization, non-linearity, and convolutional layer. Bottleneck units employ the same pipeline, but in the convolutional layer $(1, 1)$ kernels are used and the number of output channels is equal to the number of the dense block input channels.

DenseNet exploits the feature maps concatenations when transmitting information from one dense unit to another, which enables more information and details to be caught. Furthermore, each bottleneck unit performs some sort of dimensionality reduction over the feature maps, choosing the best ones.

Original DenseNet implementation has 3 dense blocks, with 5 dense units in each block, but we found that reducing the number of dense units in each block to 4 im-

proves network performance. Furthermore, we added 2 fully connected dense (ReLU) layers on the top of the whole net. Network structure is depicted on the Fig.1.

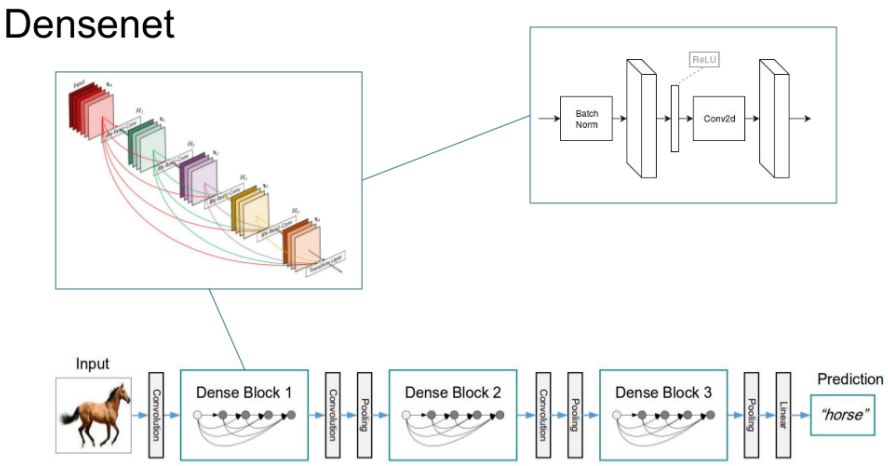

Figure 1: DenseNet architecture [2]

We trained DenseNet on the parameter setting that proved best during experiments:

- 20 epochs
- batch size = 10
- ADAM optimizer with learning rate = $1e-4$, $2e-6$ after 4-th epoch and $2e-7$ after the 5-th epoch.

3.3 CapsNet

The last but not the least model we tried is Capsule Neural Network – an architecture recently proposed by Sabour et al.[4], that shows state-of-the-art results on classical MNIST dataset. We tried to apply it to the task of brain tumor classification,

The architecture of the network is depicted on Fig. 2. It consists of two major parts: classifier and decoder. Below we describe construction and purpose of both of them.

Classifier:

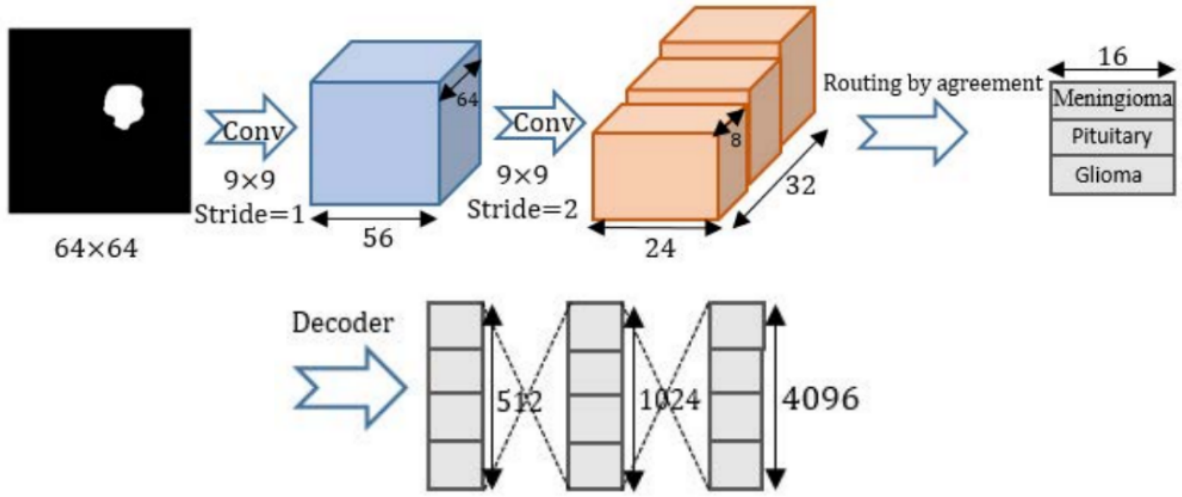

Figure 2: Capsule network architecture [1]

1. Convolution layer with 256 output channels and ReLU activation function;

2. 32 primary capsules; each of them is a separate convolution layer with 8 output channels. Outputs of the primary capsules are then stacked together and reshape so that we have $24 \cdot 24 \cdot 32$ vectors u_i of length 8, $i = \overline{1, (24 \cdot 24 \cdot 32)}$.

3. To each of those vectors we apply a squash function $squash(v) = \frac{\|v\|^2}{1+\|v\|^2} \cdot \frac{v}{\|v\|}$ to ensure that norm of each vector is between 0 and 1

4. For each vector u_i and each type of tumor we calculate vector $\hat{u}_{i|j} = W_{ij} u_i$ that is a predicted vector of class j given vector i, and each W_{ij} is a 16×8 matrix of trainable weights.

5. Finally, the output vectors for each class j is calculated via weighted sum $s_j = squash(\sum_i c_{ij} \cdot \hat{u}_{i|j})$, where scalar weights c_ij are calculated using 'Routing by agreement' algorithm proposed in [4].

6. We calculate norm of each output class vector and apply softmax function to the vector of norms. This vector is then used to calculate *margin loss*. As a label for each data sample we choose a class whose output vector has the highest norm. Then

Decoder: Once we calculated output vectors for each class and chose the classification label, we set all other output vectors but the chosen one with zeros and feed into decoder. Decoder consists of three fully connected layers and aims to reconstruct

Table 1: Brain tumor classification with different approaches

	# of parameters	F1* Score	Time to train
Vanilla CNN	**43.807.331**	**74,94%**	**∼13 sec**
DenseNet	**49.264.917**	**90.94%**	**∼4 min**
CapsNet	**57.211.776**	**91.69%**	**∼11 min**

the initial image x_i fed into the network. This reconstruction is used to calculate *reconstruction loss* which is claimed to work as a regularizer for capsule networks.

$$\text{reconstruction loss} = MSE(x_i, \text{reconstruction})$$

The total loss is a sum of two terms:

$$\text{total loss} = \text{margin loss} + \alpha \cdot \text{reconstruction loss}$$

All the trainable parameters of network are learned to minimize this total loss.

We trained capsule network on the parameter setting that prover best during experiments:

- 20 epochs
- batch size = 5
- ADAM optimizer with learning rate = $1e-3$

4 Results comparison

* - macro-averaging is used

We also found that in spite of almost the same F1 scores of DenseNet and CapsNet (Table 1.), DenseNet has 17% glioma tumors classified as meningioma. In comparison, CapsNet miss classifies only 5% gliomas as meningiomas. That difference is important, because glioma is a very dangerous type of tumor and it is important to detect it precisely.

5 Conclusion

Results of this projected confirmed CapsNets' potential in the field of medical image processing, and brain tumor classification specifically. In addition, we observed the CapsNet's ability to achieve the same F1-score as DenseNet with only 65% of the training data. However, CapsNets occupy more memory and more computationally extensive in comparison with Convolutional Nets.

Source code is available on: github.com/Dolorousrtu/capstumor

6 Contribution

Satyarth: Data preprocessing, ConvNet implementation and tuning, results comparison.

Artem: Data Augmentation, DenseNet implementation and tuning, results comparison.

Artur: Capsule network implementation and tuning, results comparison.

References

[1] Parnian Afshar, Arash Mohammadi, and Konstantinos N Plataniotis. Brain tumor type classification via capsule networks. *arXiv preprint arXiv:1802.10200*, 2018.

[2] Laurens van der Maaten Kilian Q. Weinberger Gao Huang, Zhuang Liu. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993v5*, 2017.

[3] B.A. Landman D. Fabbribi J.S. Paul, A.J. Plassard. Deep learning for brain tumor classiffication. *Proceeings of Spie*, 2017.

[4] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869, 2017.

[]