Convolutional neural networks for classification of transmission electron microscopy imagery

Sergii Gryshkevych

September 8, 2016

Contents

Abstract

An abstract.

1 Introduction

This is my thesis job report.

2 Background

2.1 Problem description

2.1.1 Lamellarity

Describe what is lamellarity and give references.

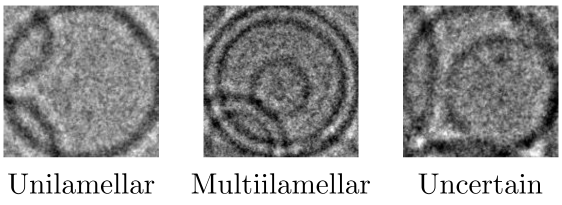

Figure 1: Lamellarity problem, 3 classes

2.1.2 Encapsulation

Describe drug delivery problem and give references.

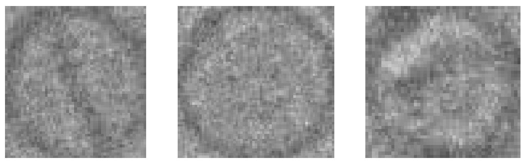

Figure 2: Encapsulation problem, 3 classes

2.2 The data set

Table 1: Lamellarity problem

Unilamellar	12368	87.29%
Multilamellar	1717	12,12%
Uncertain	84	0,59%

Table 2: Encapsulation problem

Full	24255	97.34%
Uncertain	502	2.01%
Empty	161	0.65%

Here comes description of the data set. Describe images and features.

2.2.1 Principal component analysis

Features included in principal component analysis (PCA):

- Area
- Circularity
- Perimeter
- Length
- Maximum width
- Signal to noise
- M20
- M02
- M30
- M03

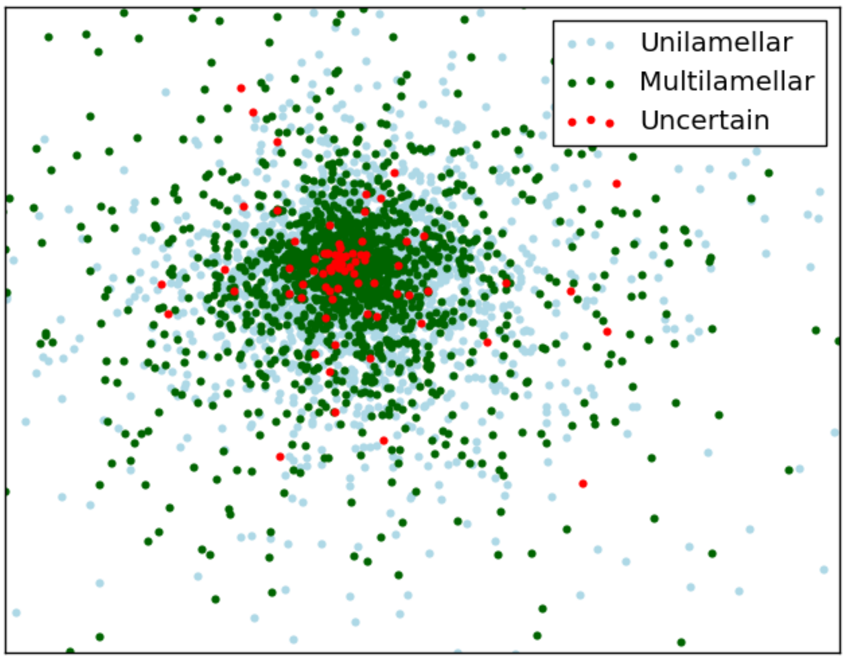

Figure 3: PCA lamellarity problem.
Explained variance ration by first two components: 94.78%, 3.39%

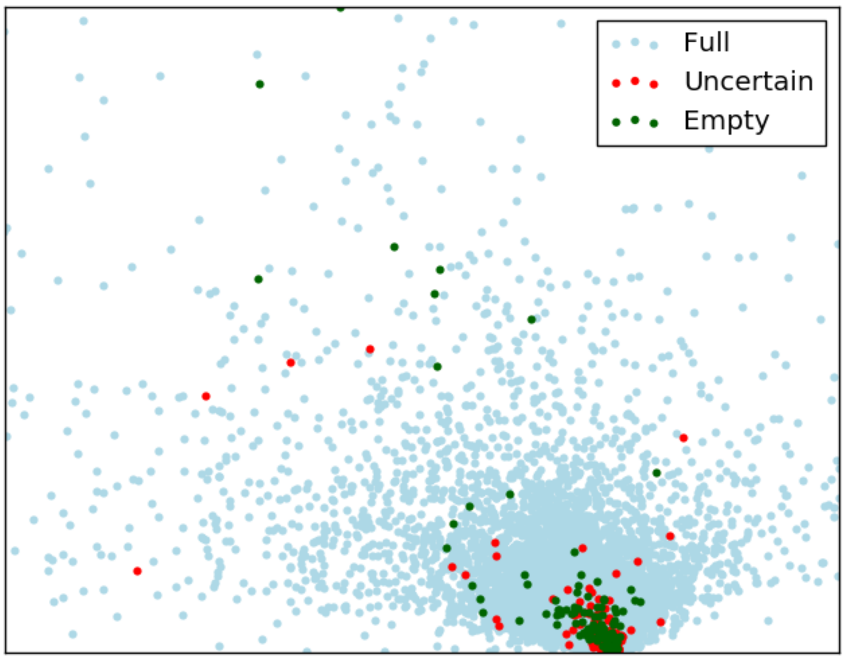

Figure 4: PCA encapsulation problem.
Explained variance ration by first two components: 66.20%, 19.96%

2.3 Imbalance problem

Describe imbalance problem. Imbalance problem has been studied in [**imbalanced-data**], ... Give examples of domains where data is usually imbalanced.

3 Methodology

3.1 Convolutional Neural Networks

Motivate use of convolutional neural networks, add a lot of references to [1]

3.2 Regularization

Motivate need of regularization and describe problems that is solves. Reularization techniques that I have tried.

3.2.1 Weight decay

☞ With weight decay many convolution kernels are equal to zero.

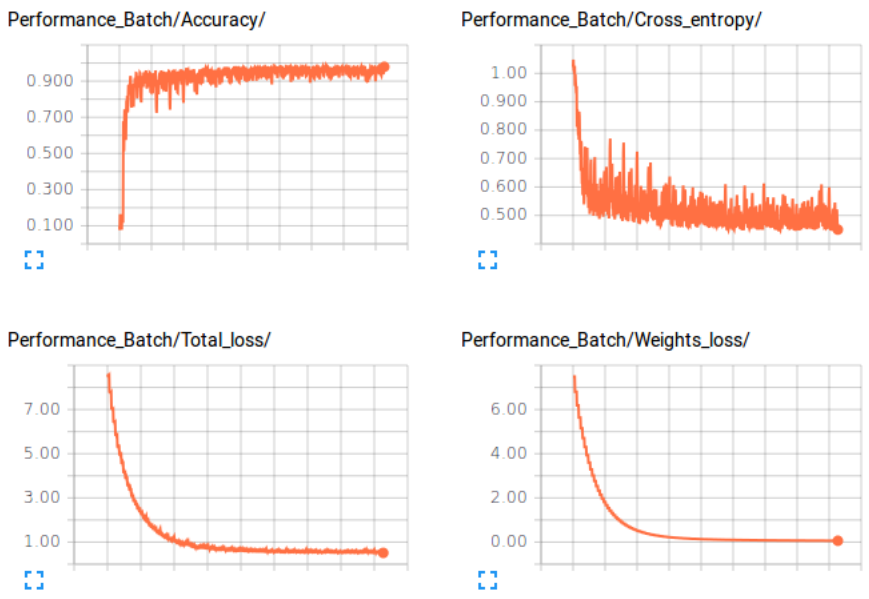

Figure 5: Batch performance metrics

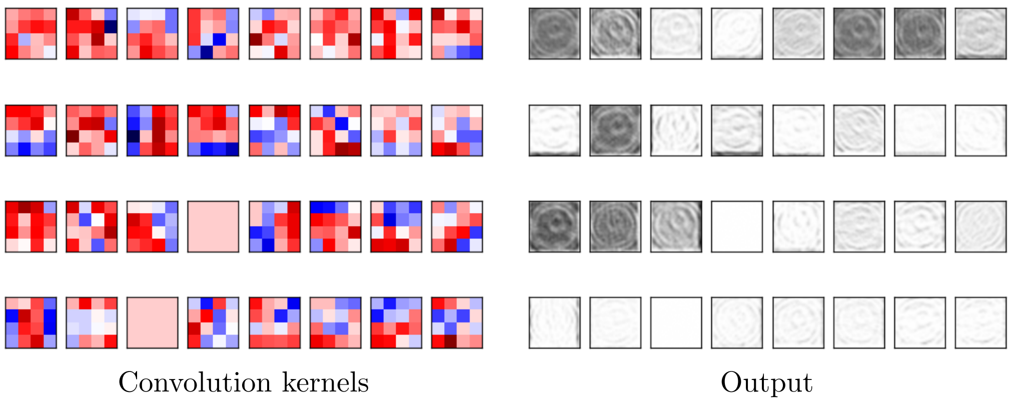

Figure 6: First convolutional layer

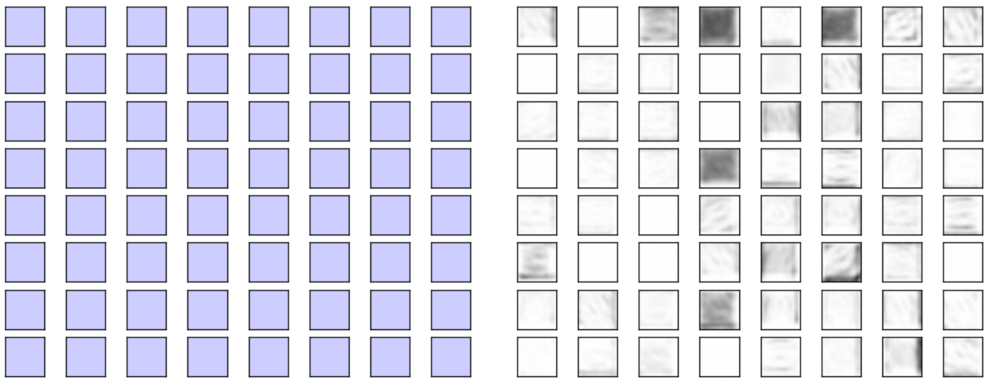
Convolution kernels, channel 27
Output (all 32 channels applied)

Figure 7: Second convolutional layer

3.2.2 Noise injection

$$x \leftarrow x\,(1-\epsilon) + (1-x)\,\frac{\epsilon}{k-1} \tag{1}$$

3.3 Data augmentation

Motivate data augmentation and give references.

3.3.1 Oversampling

Minority classes have been oversampled in the following way:

- Rotation by 90, 180 and 270 degrees
- Flipping
- Rotation of flipped image by 90, 180 and 270 degrees

One flipping and six rotations increase number of samples by factor 7. Rotation values have been selected as multiples of 90° for performance reasons. Rotation by multiples of 90° does not require any interpolation and can be performed efficiently.

3.3.2 Undersampling

Majority class has been undersampled by 20%. This means that during each epoch 20% of randomly selected examples of majority class are ignored during training.

3.3.3 SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) is an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement [2]. Synthetic examples are generated by combining each minority class example with its randomly selected k nearest neighbors, where k is a parameter that depends on amount of required over-sampling.

☞ Add figure of some liposome, its nearest neighbors and generated synthetic examples.

3.3.4 Synthetic data

Synthetic and semi-synthetic alternative to real data have been discussed and evaluated in [3]. I also tried to generate synthetic data, train network on it and evaluate on real data. Performance was poor, much worse than training on over-sampled and semi-synthetic data produced by SMOTE.

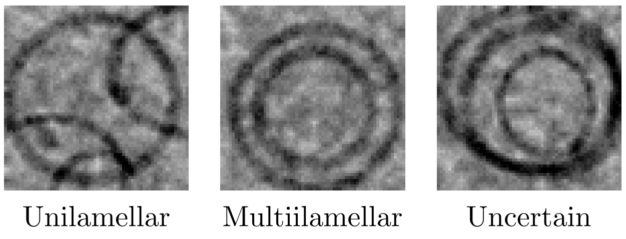

Figure 8: Synthetic examples, lamellarity problem

☞ Kommentar från Max: Det kan vara svårt att motivera en syntetiseringsmetod som inte grundar sig på statistiska mått på huvuddatan. Prova gärna att göra mindre rotationer, 10 grader eller så, använd gärna Lanczos-interpolering, du kan fråga Ida-Maria om referens för det (poängen är att Lanczos bibehåller textur-statistik mycket bättre än t ex linjärinterpolering).

3.4 Loss function

☞ This section is taken from project report for Machine Learning course. How to cite it?

The multi-class logarithmic loss function is a loss function that represents the price paid for inaccuracy of predictions in classification problems [4]. The formula is:

$$loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j} log(p_{i,j}) \tag{2}$$

Where

- N – number of observations
- M – number of classes, it is five in our case
- *log* – natural logarithm
- $y_{i,j}$ – is 1 if observation i is in class j and 0 otherwise
- $p_{i,j}$ – is the predicted probability that observation i is in class j

In order to calculate multi-class logarithmic loss the classifier must assign a probability to each class rather than simply yielding the most likely class. This fact constitutes the main difference between accuracy and logarithmic loss. In order to consider a prediction as accurate, it is sufficient that classifier marks the correct class as the most likely one, no matter how confident the prediction is.

Figure 9 shows log loss from a single class where predicted probability ranges from 0 (the completely wrong prediction) to 1 (the correct prediction). As predicted probability moves closer to 1, log loss decreases gently to 0. On the contrary, log loss increases rapidly as predicted probability moves towards zero. It is clear from Figure 9 that the multi-class logarithmic loss heavily penalizes classifiers that are confident about an incorrect classification.

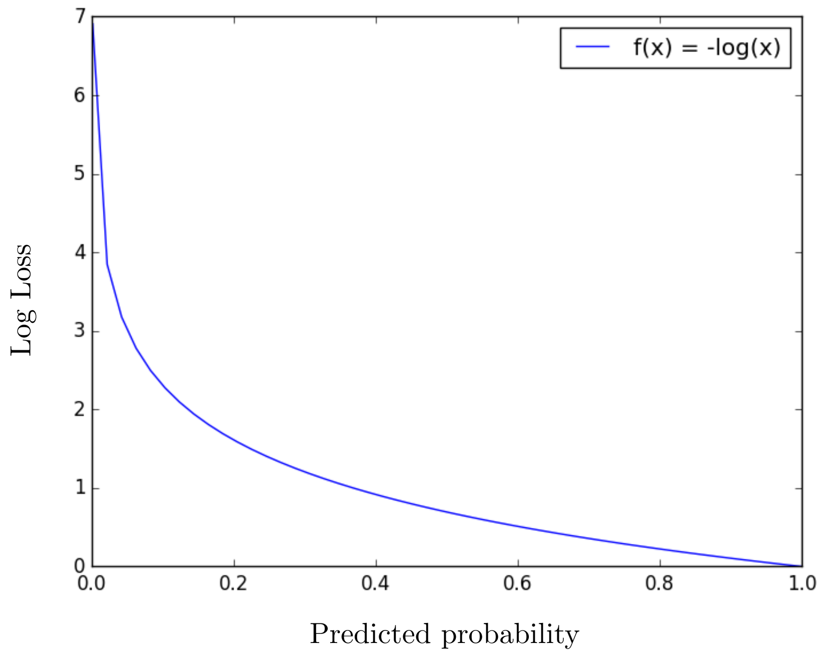

Figure 9: Log loss of a single class where predicted probability ranges from 0 to 1.

Consider an example of a binary classifier and let us take a look at the effect of various predictions for class membership probability.

- Classification that assigns equal probabilities to both classes results in loss $-log(0.5) = 0.6932$.
- Classification confident in the correct class results in loss $-log(0.9) = 0.1054$.
- Classification confident in the wrong class results in loss $-log(0.1) = 2.3026$.

In other words, the log loss function encourages moderate predictions. It is better to make all classifications neutral rather than make 50% correct classifications and 50% completely wrong predictions.

3.4.1 Dropout

3.4.2 Early stopping

4 Implementation of CNN

4.1 Network visualization

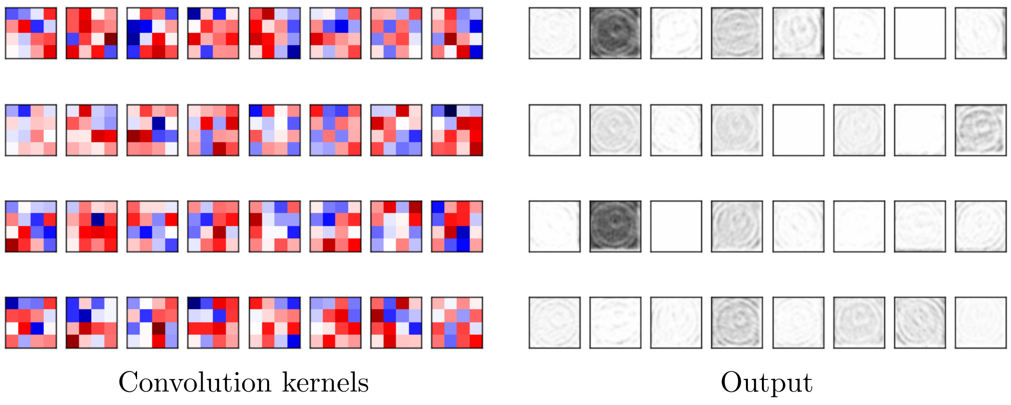

Figure 10: First convolutional layer

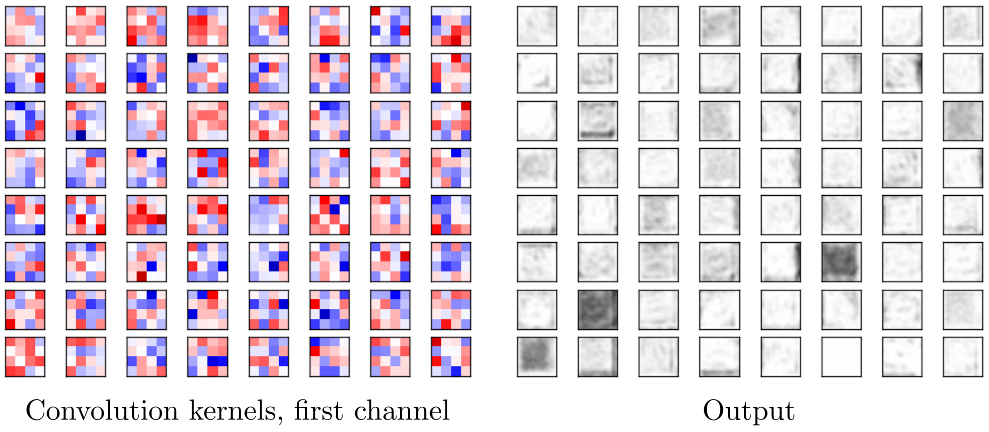

Figure 11: Second convolutional layer

4.2 Review of deep learning software libraries

Deep learning software tool have been benchmarked in [**benchmarking˙dl˙tools**]

Table 3: Comparison of characteristics of deep learning frameworks

	Linux		Windows		Mac OS		Language bindings		
	CPU	GPU	CPU	GPU	CPU	GPU	Python	C++	License
Caffe	✓	✓	✓	✓	✓	✓	✓	✗	BSD 2
cntk	✓	✓	✓	✓	✗	✗	✓	✓	Custom[1]
Keras	✓	✓	✗	✗	✓	✗	✓	✗	MIT
Lasagne	✓	✓	✓	✓	✓	✓	✓	✗	MIT
mxnet	✓	✓	✓	✓	✓	✓	✓	✓	Apache 2.0
nolearn	✓	✓	✓	✓	✓	✓	✓	✗	MIT
Tensorflow	✓	✓	✗	✗	✓	✗	✓	✓	Apache 2.0
Theano	✓	✓	✓	✓	✓	✓	✓	✗	BSD 3

Table 4: Comparison of popularity of deep learning frameworks

	Github				Stackoverflow
	Contributors	Stars	Forks	Subscribers	Questions
Caffe	208	12349	7513	1559	1038
cntk	77	6181	1343	688	12
Keras	257	8044	2470	639	444
Lasagne	51	2452	673	202	153
mxnet	174	4969	1887	559	22
nolearn	9	718	203	52	44
Tensorflow	380	31445	13414	2983	3275
Theano	251	4475	1609	424	1436

[1] Allows commercial use reference

5 Evaluation

5.1 ROC

5.2 Confusion matrix

☞ Problem with **Uncertain** class.

Table 5: Lamellarity problem

	Unilamellar	Uncertain	Multilamellar
Unilamellar	0.97	0.00	0.03
Uncertain	0.29	0.00	0.71
Multilamellar	0.04	0.00	0.96

Table 6: Packiging problem

	Empty	Full	Uncertain
Empty	0.41	0.41	0.18
Packed	0.00	0.95	0.05
Uncertain	0.20	0.11	0.69

5.3 K-fold cross validation

6 Results

7 Conclusion

8 Discussion

References

[1] Ian Goodfellow Yoshua Bengio and Aaron Courville. "Deep Learning". Book in preparation for MIT Press. 2016. URL: `http://www.deeplearningbook.org`.

[2] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *J. Artif. Int. Res.* 16.1 (June 2002), pp. 321–357. ISSN: 1076-9757. URL: `http://dl.acm.org/citation.cfm?id=1622407.1622416`.

[3] Carolina Wählby Omer Ishaq Vladimir Curic. "Training of macine learning methods for fluorescent spot detection".

[4] Lorenzo Rosasco et al. "Are Loss Functions All the Same?" In: *Neural Comput.* 16.5 (May 2004), pp. 1063–1076. ISSN: 0899-7667. DOI: `10.1162/089976604773135104`. URL: `http://dx.doi.org/10.1162/089976604773135104`.