# COMSM0045: PRACTICAL-Lab4

Dima Damen
Dima.Damen@bristol.ac.uk

Bristol University, Department of Computer Science
Bristol BS8 1UB, UK

November 12, 2020

## What else can we do?

- Data Augmentation
- Debugging Strategies
- Dropout

## Data Augmentation

- Data augmentation is making the most of the training samples by introducing variations of these samples to accommodate for required invariances
- Why Data Augmentation?
  - Because it's all about the size of your data –> More data for training
  - **More importantly...** to accommodate invariances

## Invariances in data

- A *problem* is **invariant** to a *property* when the problem remains unchanged when transformations of a certain type are applied.

| **Problem** | **Invariant to...** |
|---|---|
| Object Recognition | translation, rotation, scaling, viewpoint |
| Number plate recognition | translation, scaling |
| Action Recognition | translation, rotation, scaling, viewpoint, speed |

## Invariances in data

| **Problem** | **Invariant to...** |
|---|---|
| Object Recognition | translation, rotation, scaling, viewpoint |

- How to provide invariance? $\rightarrow$ *artificially* augment for:
  - Translation: shifts – luckily CNNs do that for us
  - Rotation: rotations
  - Scaling: croppings
  - Viewpoint: Minor - affine transformations, otherwise :-( collect more data!

## Invariances in data

- Other invariances:
  - invariance to random noise
  - invariance to occlusion
  - invariance to lighting conditions
  - invariance to colour variations
  - invariance to time of year!? — Generative!

# Data Augmentation

- Data augmentation for invariances existed before deep learning

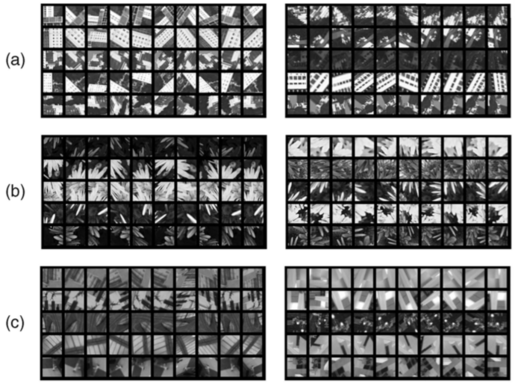

Fig. 7. Warped patches from the images of Fig. 6 show the range of

Ozuysal et al (2010). Fast Keypoint Recognition Using Random Ferns. TPAMI.

## Data Augmentation

- Why can’t current deep learning methods do that automatically for us?

## But... Do I need more Data?

- There is a balance between the expense of collecting labelled data and refining the method
- How are you performing on your training data?
  - poorly $\rightarrow$ your algorithm needs work, you are not making the most of the data you have
  - quite well $\rightarrow$ How are you performing on your test data?
    - poorly $\rightarrow$ try augmentation, otherwise collect more data
    - quite well $\rightarrow$ you're done! Find a more interesting problem
- I collected more data but things did not improve?
  - Think about the *quality* of your data
  - Think about the *quality* of your *labels*
  - $\rightarrow$ fix and start over

## Debugging Deep Learning Algorithms

- When a general machine learning code performs poorly, including deep learning code, it is very tricky to decide whether that is a bug in the code or a problem in the algorithm
- Compiling correctly and getting numbers out is not an indication of correctness
- We do not know what the "correct" implementation will give in terms of accuracy, that is in fact what we wish to discover
- Careful debugging is thus a must

## What is your baseline performance?

- Remember: you cannot perform worse than the baseline!
- What can an algorithm that makes decisions by “chance” do?
- For a binary classifier, your baseline is 50%
- For a classifier into $N$ balanced classes, your baseline is $\frac{1}{N}\%$
- For a classifier with unbalanced classes??

## Debugging Strategies

- *Mickey Mouse Examples* - test your solution on small tests that you know the outcome for
- Evaluate the performance of your building blocks in isolation
- Monitor the model in action
- Look at failure cases (qualitative assessment)
- Checkpoints and model saving

## Debugging Strategies

*"Directly observing the machine learning model performing its task will help to determine whether the quantitative performance numbers it achieves seem reasonable"*

Goodfellow et al, p432

## Debugging Strategies

*“Evaluation bugs can be some of the most devastating bugs because they can mislead you into believing your system is performing well when it is not”*

Goodfellow et al, p432

## Dropout

- What is regularisation?
- Remind yourself about dropout, as a regularisation/ensemble approach, from the lectures.

## By the end of these lab sessions, you should be able to...

- Define a Fully-Connected Deep Neural Network (DNN) architecture
- Define a shallow Convolutional Neural Network (CNN) architecture
- Train and validate a CNN, and monitor its progress and results using Tensorboard
- Understand and estimate the effect of changing hyper-paramters on your results
- Implement and evaluate a variety of data augmentation techniques
- Implement dropout as one of the most common regularisation approaches

## And now....

**READY....**

**STEADY....**

**GO...**