Deep Learning

Krystian Mikolajczyk & Carlo Ciliberto

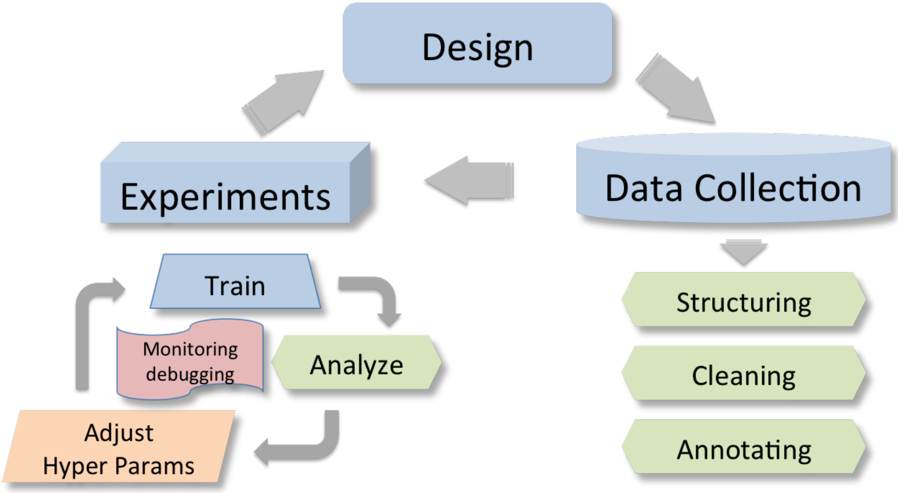
Department of Electrical and Electronic Engineering

Imperial College London

Practical development process

- System design/choice
- Data collection and augmentation
- Hyper parameters search
- Monitoring and debugging

Development process



Design

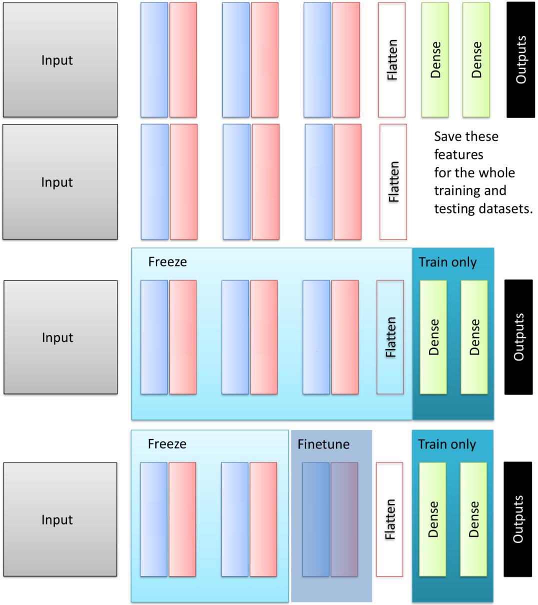
Baseline: end-to-end model

- Was the task studied before? Do literature review!
  - Start from competitions/survey papers
  - Establish a reasonable end-to-end system
- Choose the general category of model based on the structure of your data:
  - MLP for fixed size vectors
  - CNN for images
  - RNN for sequential data
- Nonlinear activations
  - Avoid sigmoid (except for output)
  - ReLU preferred (possibly Leaky ReLU)
  - Use Maxout if most ReLU units die (have zero activation)
- Weights & Biases
  - Random initialization with proper variance
  - For ReLU we prefer a small positive bias to activate ReLU

Design

Finetuning - borrow knowledge

- Pretrain your NN on a large dataset (e.g. same modality, similar task)
  - or start from a pretrained NN
- Option 1: remove / reshape the last few layers and use the features
- Option 2: Fine-tune the parameters on your own dataset
  - Freeze the parameters of first few layers, or make the learning rate small for them
  - Small data - train last FC layers only
  - Medium data - can finetune other layers
  - Use only 1/10th of the original learning rate in finetuning top layer, and 1/100th on intermediate layers



Data Collection

Study the application domain!

Collect data for the task

- How much data to collect?
  - The more the better
  - Depends on the effect we want to observe
  - Required error bounds and accuracy
- How to label the data?
  - Mechanical Turk, Freelancer, experts, ...
- Avoid bias
  - Selection, Sampling, ...

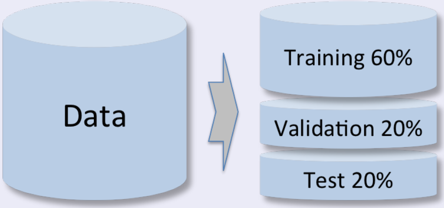
Dataset preparation/curation

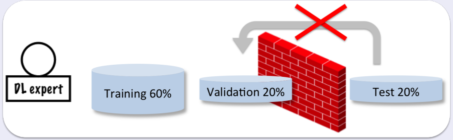
- Data structuring & formatting
  - Is the data format suitable?
  - Standardization
- Data cleaning
  - incomplete data, anonymization, missing annotation, correction ...
- Data normalization
  - Value clipping/normalization
  - Whitening

Data Collection

Data split

- Training set
  - typically 60%, to run the learning algorithm on
  - Keep training data balanced
- Validation set
  - typically 20%, to tune hyper parameters, select features
  - make other decisions regarding the learning algorithm
  - also called development set
- Test set
  - typically 20%, to evaluate the performance of the algorithm





Testing

Do not use test data to make any decisions to improve learning!

Validation

Regularisation

$$\mathcal{L}_n(h) = \widehat{R}_n(h) + \lambda \underbrace{\Omega(h)}_{overfit\ \ penalty}$$

Validation

$$\underbrace{\mathcal{L}_n(h)}_{direct\ \ estimation} = \widehat{R}_n(h) + \lambda \underbrace{\Omega(h)}_{overfit\ \ penalty}$$

1. Split the training data $\mathcal{D}$ into training $\mathcal{D}_{train}$ and validation $\mathcal{D}_{val}$ sets.
2. Train g on $\mathcal{D}_{train}$.
3. Estimate its performance on $\mathcal{D}_{val}$ ($v = |\mathcal{D}_{val}|$):

$$\check{R}_v(g) = \frac{1}{v} \sum_{(x_i, y_i) \in \mathcal{D}_{val}} \ell(g(x_i), y_i)$$

Very good estimate of $R(g)$

$$\mathbb{E}_{\mathcal{D}_{val}}\left[\check{R}_v(g)\right] \approx R(g) \leqslant \check{R}_v(g) + \underbrace{\Omega(v, \delta)}_{\sim\sqrt{\log(1/\delta)/v} \leftarrow\ one\ \ model\ \ only\ \ on\ \ v-points} \quad \text{w. p. } 1 - \delta$$

- D_{val} is unbiased, small Hoeffding bound, only one g is considered.
- Select $\lambda^* = \text{argmin}_\lambda \check{R}_v(g)$, then train on the whole $\mathcal{D}$ with λ^*.

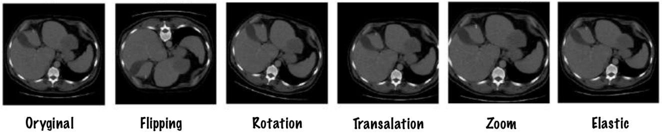
Data Augmentation

- Image data augmentation
- Text data augmentation
- Audio data augmentation

Data Augmentation

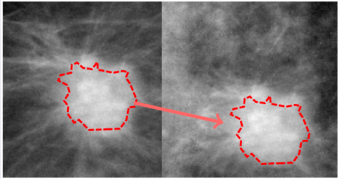
Image Data Augmentation

- Adding noise
- Generating modified samples




```
datagen = ImageDataGenerator(rotation_range=25,
                             shear_range=0.41,
                             horizontal_flip=True,
                             vertical_flip=True,
                             fill_mode='reflect',
                             spline_warp=True,
                             warp_sigma=10,
                             warp_grid_size=3)
```

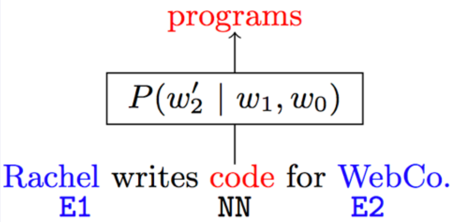
- Medical data
  - Segment tumor mass
  - Move
  - Resample background tissue
  - Blend



Data Augmentation

Text Data Augmentation

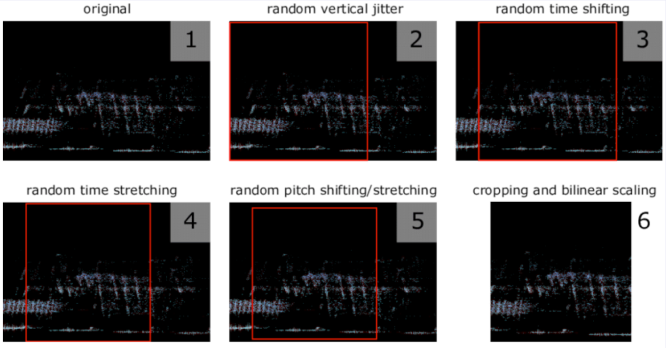
- Adding noise
- Inserting synonyms
- A conditional word-swap with externally trained language model and specifically targeting nouns (NN) between entity mentions (E1,E2)
- Rare words in new, synthetically created contexts



Data Augmentation

Audio Data Augmentation

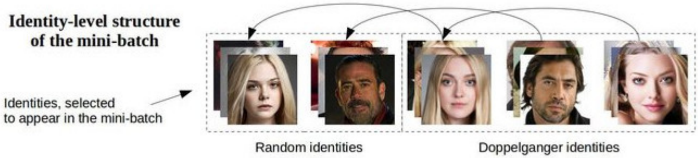
- Adding noise
- Vertical jitter
- Time shifting
- Time stretching: change the speed of the audio signal
- Pitch shift
- Cropping and bilinear scaling



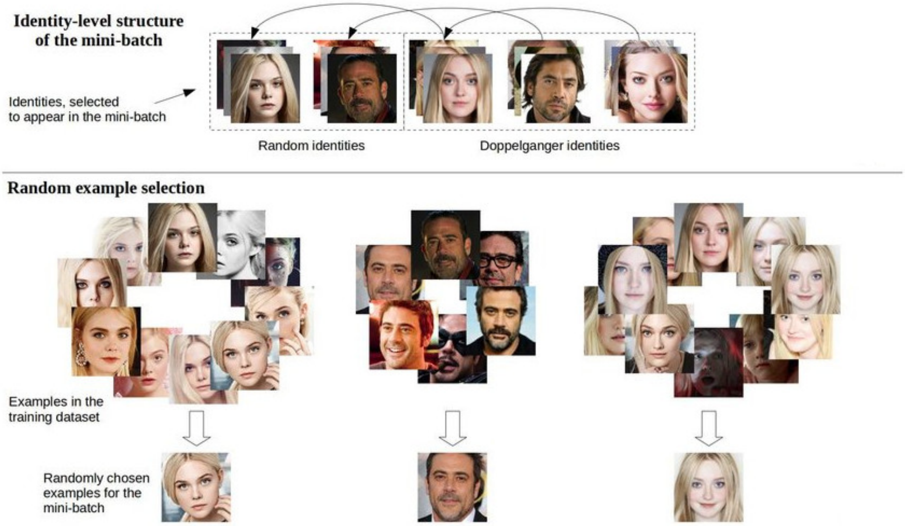
Hard Positive/Negative Mining

Adversarial training

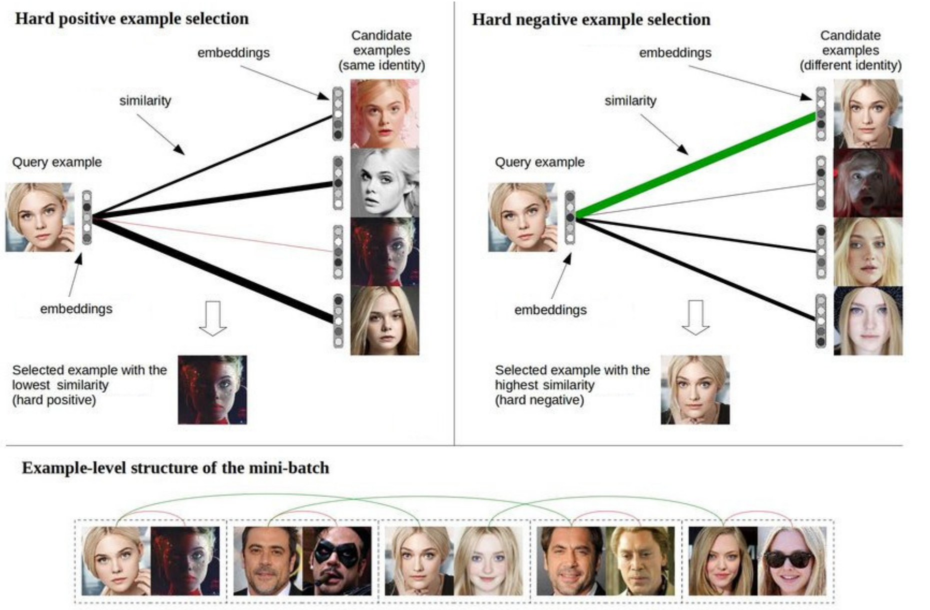
- Train the system from randomly formed mini-batches with balanced positive/negative examples
- Identify hard examples (close to decision boundary) during validation
- Form a new mini-batch by including the hard examples from the the previous iteration.



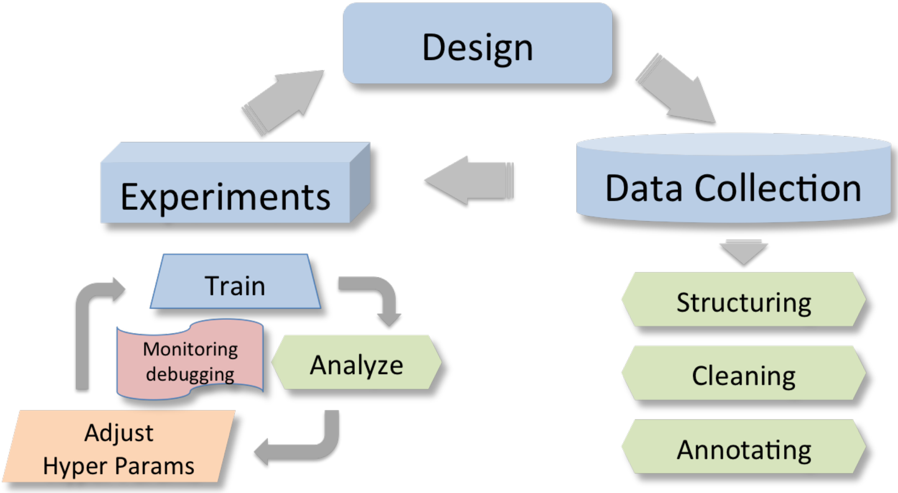
Hard Positive/Negative Mining



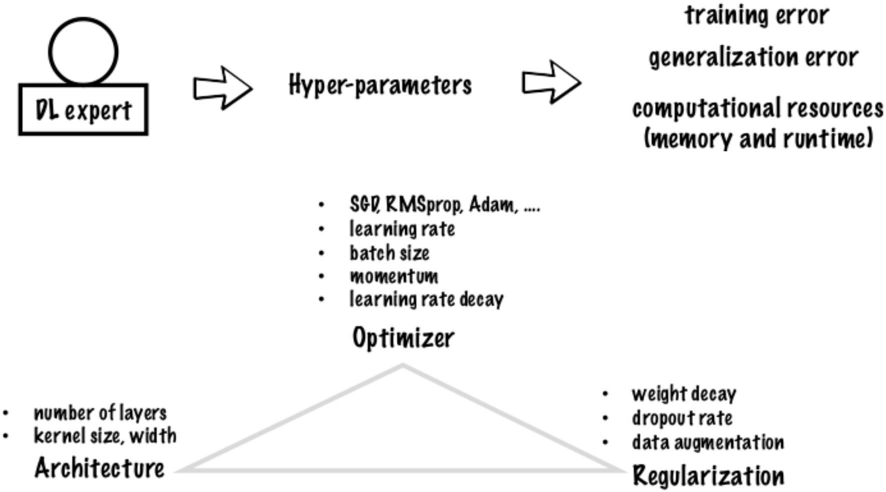
Hard Positive/Negative Mining



Development process



Parameter setting



Parameter setting

Hyper-parameters

- Optimization of hyper-parameters affects the quality of local minima that our model can reach (effective model capacity)
  - memory problems? reduce batch size
- The bigger the architecture the higher the model's capacity
  - caveat: memory and computation time are limited
- To decrease generalization gap increase regularization
  - increase weight decay and dropout rate to reduce model capacity
  - data augmentation does not affect model capacity

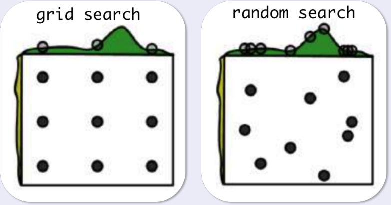
Good rule of thumb

Start from default parameters!

Parameter setting

Hyper parameter search: Good intuition and experience

- Learning Rate / Momentum
  - Decrease learning rate while training
  - Typical momentum to 0.8 - 0.9
- Batch Size
  - Shuffle data before batching
  - For large dataset: set to whatever fits your memory
  - For smaller dataset: find a tradeoff between instance randomness and gradient smoothness
- More efficient to optimize hyper-parameter with randomly chosen trials rather than on grid
- Model based hyper-parameter selection
- Use coarse to fine search for hyper-parameters
- Search on log scale (eg. learning rates)



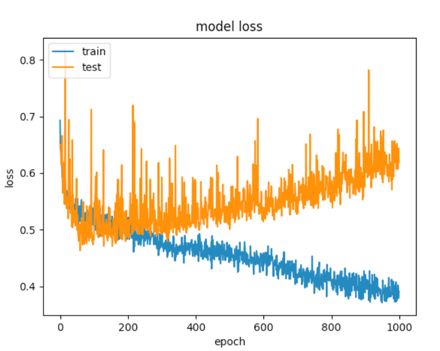
Monitoring and debugging

- Set a small dataset
- Reasoning using train and validation error/plots
- Monitoring histograms of activations and gradient updates
- Analysing of model predictions and errors

Monitoring and debugging

Set a small dataset

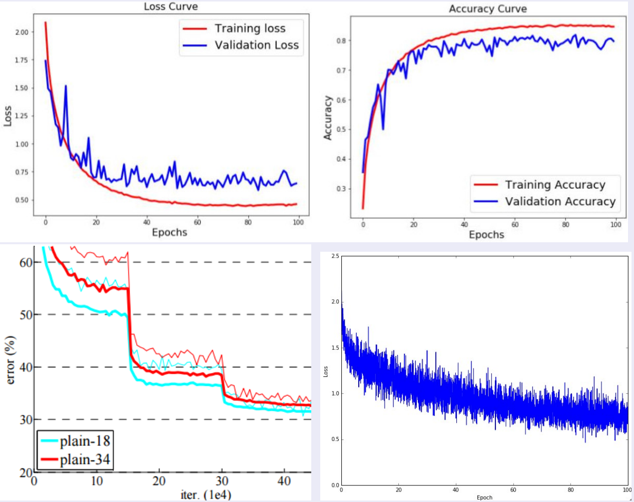
- Observe training error/plot
  - The model should overfit
- Change the optimisation parameters if the model does not overfit
  - If unsuccessful, inspect the code for potential bugs
- Reduce the size of the model (architecture) if model compilation time is long



Monitoring and debugging

- Optimize your hyper-parameter in validation and evaluate on test
- Keep track of training and validation loss during training
- Do early stopping if training and validation loss diverge
- Use patience - wait this long from last change
- Relative improvement threshold (significance)
- Loss does not tell you all. Try precision, class-wise precision, and other metrics

Train and validation error/plots



Monitoring and debugging

Visualize model predictions

- Check input data and annotations
- Check if predictions make sense

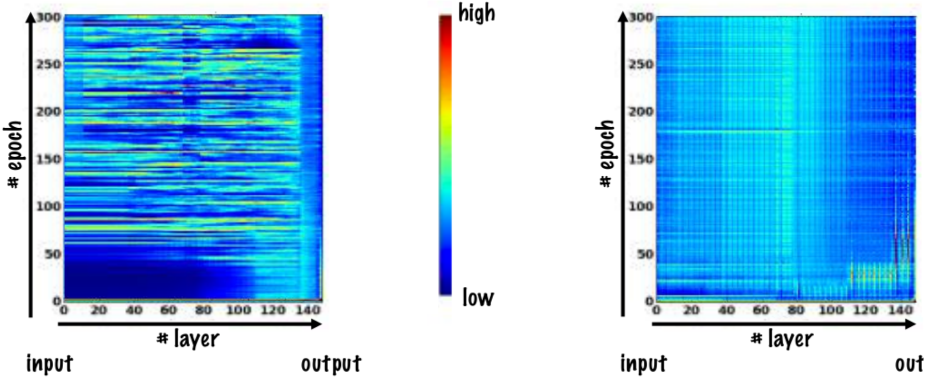
Understanding the underlying causes of the errors

- Get a sample of e.g. 100 validation set examples for which the system failed
- Examine these examples manually/visually
- Identify the most common errors
  - A simple excel table might be enough

Monitoring and debugging

Monitor histograms of activations and gradient updates per layer

- Erratic/unstable
- Uniform



Monitoring and debugging

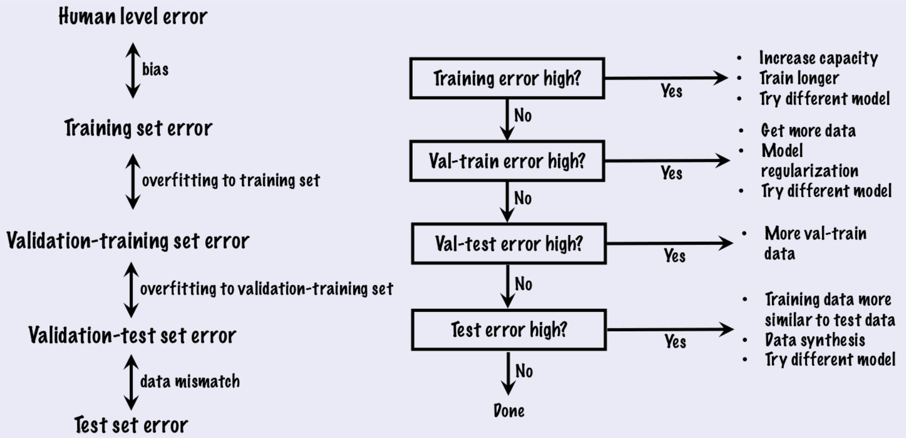
Generalisation problem

- Biased data distributions (not i.i.d.)

- Mix datasets and test
- Split validation into training validation and test validation

Monitoring and debugging

Error analysis



Summary

- System design/choice
- Data collection and augmentation
- Hyper parameters search
- Monitoring and debugging