EE3-25: Deep Learning

Krystian Mikolajczyk & Carlo Ciliberto

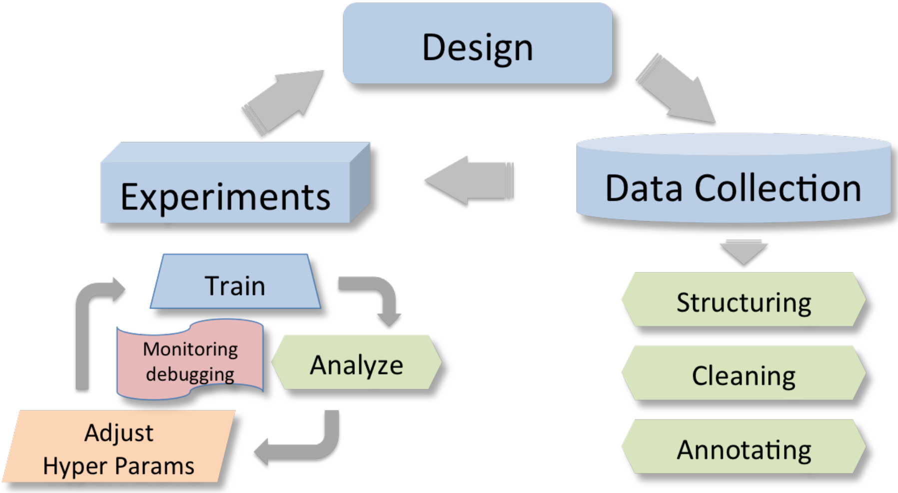
Department of Electrical and Electronic Engineering

Imperial College London

Practical development process

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

Development process



Design

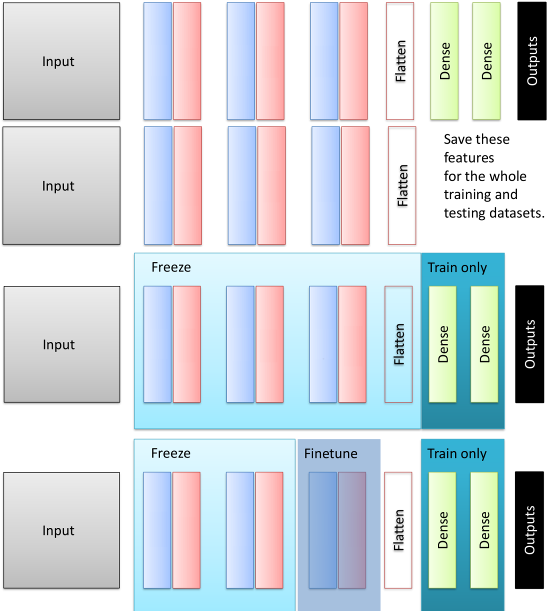
Baseline: end-to-end model

- 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

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, ...

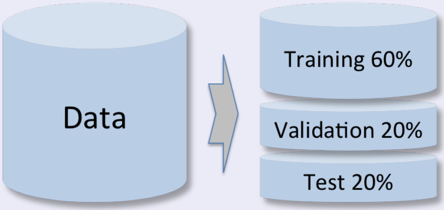
Dataset preparation/curation

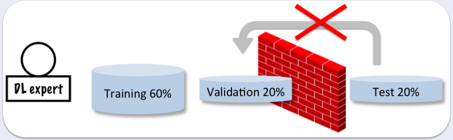
- Data structuring & formatting
  - Is the data format suitable?
  - Standardization
- Data cleaning
  - incomplete data, anonymization, missing annotation, correction ...
- Data normalization
  - Value clipping/normalization
  - Whitening

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^*.

Validation: More generally

Validation

Given hypothesis classes $(\mathcal{H}_1, \lambda_1), \ldots, (\mathcal{H}_i, \lambda_i), \ldots, (\mathcal{H}_M, \lambda_M)$,

1. Split training data $\mathcal{D}$ into $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ sets.
2. Train g_i on $\mathcal{D}_{train}$, $\quad |\mathcal{D}_{train}| = n - v$
3. Select i_* such that $g_{i_*} = \text{argmin}_i \check{R}_v(g_i)$ on $|\mathcal{D}_{val}| = v$
   - i - not only regularisation, can be other hyperparameter.
4. Select $\mathcal{H}_{i_*}, \lambda_{i_*}$ and train new g on $\mathcal{D}_{val} \cup \mathcal{D}_{train}$ to get the final g_*.

- Cost? $\quad (n - v)$ and learning curves.
- if $v \uparrow$ then $\check{R}_v(g) \sim R(g) \uparrow$, but $|\mathcal{D}_{train}| = n - v \downarrow$ then $R(g) \uparrow$
- How big $|\mathcal{D}_{val}| = \frac{|D_{train}|}{5}$
- Why "validation" and not "test" set? Unlucky split of dataset $\mathcal{D}$?

Validation

Cross validation

1. Split data $\mathcal{D}$ into K disjunct parts: $\mathcal{D} = \cup_{k=1}^{K} \mathcal{D}_k$
2. For each k, create training and validation set:
   - Training $\mathcal{D}_{\bar{k}} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cancel{\mathcal{D}_k} \cup \ldots \cup \mathcal{D}_K$, $|\mathcal{D}_{\bar{k}}| = \bar{n} = \frac{K-1}{K} n$
   - Validation $\mathcal{D}_k$, $|\mathcal{D}_k| = \bar{k} = \frac{n}{K}$
3. Train $g_{\bar{k}}$ on $\mathcal{D}_{\bar{k}}$ with ERM

- Validation error of $g_{\bar{k}}$ on $\mathcal{D}_k$ is $\breve{R}_k(g_{\bar{k}}) = \frac{1}{|\mathcal{D}_k|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_k} \ell(g_{\bar{k}}(\mathbf{x}_i), y_i)$
- Cross-validation error – K-fold cross validation

$$\breve{R}_{CV} = \frac{1}{K} \sum_{k=1}^{K} \breve{R}_k(g_{\bar{k}})$$

Cross validation: Is this good?

Expected performance of the learnt hypothesis: $\mathbb{E}_{\mathcal{D}}\left[R(g^{(\mathcal{D})})\right]$

$$\mathbb{E}_{\mathcal{D}_k}\left[\check{R}_k(g_{\bar{k}})\right] = \mathbb{E}_{\mathcal{D}_k}\left[\frac{1}{|\mathcal{D}_k|}\sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}_k}\ell(g_{\bar{k}}(\mathbf{x}_i),y_i)\right]$$

$$= \mathbb{E}_{x,y}\left[\ell(g_{\bar{k}}(\mathbf{x}),y)\right]$$

$$\approx R(g_{\bar{k}}) = R(g^{(\mathcal{D}_{\bar{k}})})$$

If all chunks are of the same size, then $g_{\bar{k}}$ is trained on $n = \frac{K-1}{K}n$ points:

$$\mathbb{E}_{\mathcal{D}}\left[\check{R}_k(g_{\bar{k}})\right] \approx \mathbb{E}_{D_k}\left[R(g_{\bar{k}})\right] \approx \mathbb{E}_{\mathcal{D}}\left[\check{R}_{CV}\right]$$

Cross validation

$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\cdots$	$\mathcal{D}_K$	$\rightarrow$	$\breve{R}_1(g_{\bar{1}})$
$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\cdots$	$\mathcal{D}_K$	$\rightarrow$	$\breve{R}_2(g_{\bar{2}})$
$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\cdots$	$\mathcal{D}_K$	$\rightarrow$	$\breve{R}_3(g_{\bar{3}})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$		
$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\cdots$	$\mathcal{D}_K$	$\rightarrow$	$\breve{R}_K(g_{\bar{K}})$

Cross-validation error

$$\breve{R}_{CV} = \frac{1}{K}\sum_{k=1}^{K} \breve{R}_k(g_{\bar{k}})$$

$$\mathbb{E}_{\mathcal{D}}\left[\breve{R}_{CV}\right] \approx \mathbb{E}_{\mathcal{D}_k}\left[R(g_{\bar{k}})\right]$$

with $\bar{n} = (K - 1/K)n$:

Typical K choices:

- $K = 10$: 10-fold cross validation
- $K = n$: leave-one-out cross validation

Cross validation

K-fold Cross Validation

Given: hypothesis classes $(\mathcal{H}_1, \theta_1), \ldots, (\mathcal{H}_i, \theta_i), \ldots, (\mathcal{H}_M, \theta_M)$,

with θ – any hyperparameter

K training sets $\mathcal{D}_{\bar{k}} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cancel{\mathcal{D}_k} \cup \ldots \cup \mathcal{D}_K$, $|\mathcal{D}_{\bar{k}}| = \bar{n} = \frac{K-1}{K} n$

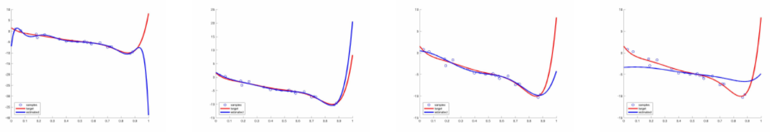
K validation sets $\mathcal{D}_k$, $|\mathcal{D}_k| = \bar{k} = \frac{n}{K}$

1. Train $g_{i,\bar{k}}$ with ERM for every $(\mathcal{H}_i, \theta_i)$ and every $\mathcal{D}_{\bar{k}}$
2. Compute $\check{R}_{CV}(g_i)$ for every g_i on all $\mathcal{D}_k$
3. Select $g_{i_*} = \operatorname{argmin}_{g_i} \check{R}_{CV}(g_i)$
4. Use $(\mathcal{H}_{i_*}, \theta_{i_*})$ and whole data set $\mathcal{D}$ to train final g_*

Regularized Loss Minimization

Regularization

$\lambda = 0$ $\lambda = 0.0001$ $\lambda = 1$ $\lambda = 100$



overfitting $\longrightarrow$ $\longrightarrow$ underfitting

Validation

$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\cdots$	$\mathcal{D}_K$	$\rightarrow$	$\check{R}_1(g_{\bar{1}})$
$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\cdots$	$\mathcal{D}_K$	$\rightarrow$	$\check{R}_2(g_{\bar{2}})$
$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\cdots$	$\mathcal{D}_K$	$\rightarrow$	$\check{R}_3(g_{\bar{3}})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$		
$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\cdots$	$\mathcal{D}_K$	$\rightarrow$	$\check{R}_K(g_{\bar{K}})$

$$\check{R}_{CV} = \frac{1}{K}\sum_{k=1}^{K}\check{R}_k(g_{\bar{k}})$$

Regularized Loss Minimization (RLM)

- Hypothesis class $\mathcal{H} = \cup_i(\mathcal{H}_i, \lambda_i)$, with $i \in \mathbb{N}$ e.g. $\lambda_i \in \{0.0001, 0.001, \ldots\}$
- Augmented error:

$$\mathcal{L}_{\bar{k}}(\mathbf{w}, \lambda) = \widehat{R}_{\bar{k}}(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

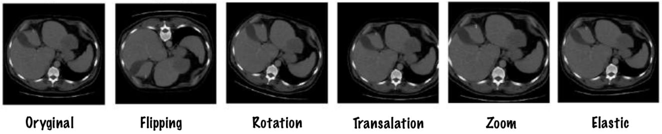
$$\text{e.g. } \Omega(\mathbf{w}) \in \{\|\mathbf{w}\|_1, \|\mathbf{w}\|_2^2, \|\mathbf{w}\|_Q^2, \ldots\}$$

- RLM solution:
  - for all i, train on $\mathcal{D}_{\bar{k}}$: $g(\mathbf{w}_{\lambda_i}) = \text{argmin}_{\mathbf{w}}\, \mathcal{L}_{\bar{k}}(\mathbf{w}, \lambda_i)$
  - from all i, select on $\mathcal{D}_k$: $g(\mathbf{w}_{\lambda *}) = \text{argmin}_{\lambda_i}\, \check{R}_k(g(\mathbf{w}_{\lambda_i}))$

Data Augmentation

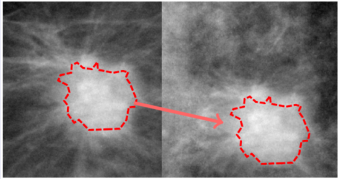
Image Data Augmentation

- 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)
```

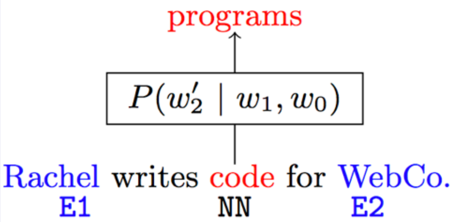
- Medical data
  - Segment tumor mass
  - Move
  - Resample background tissue
  - Blend



Data Augmentation

Text Data Augmentation

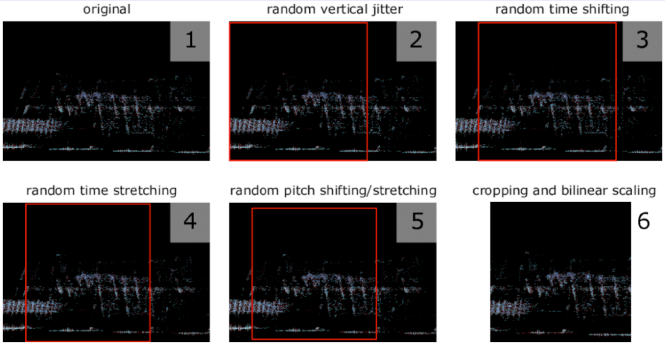
- Adding noise
- Inserting synonyms
- A conditional word-swap with externally trained language model and specifically targeting nouns (NN) between entity mentions (E1,E2)
- Rare words in new, synthetically created contexts



Data Augmentation

Audio Data Augmentation

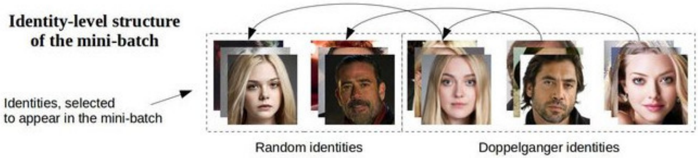
- Adding noise
- Vertical jitter
- Time shifting
- Time stretching: change the speed of the audio signal
- Pitch shift
- Cropping and bilinear scaling



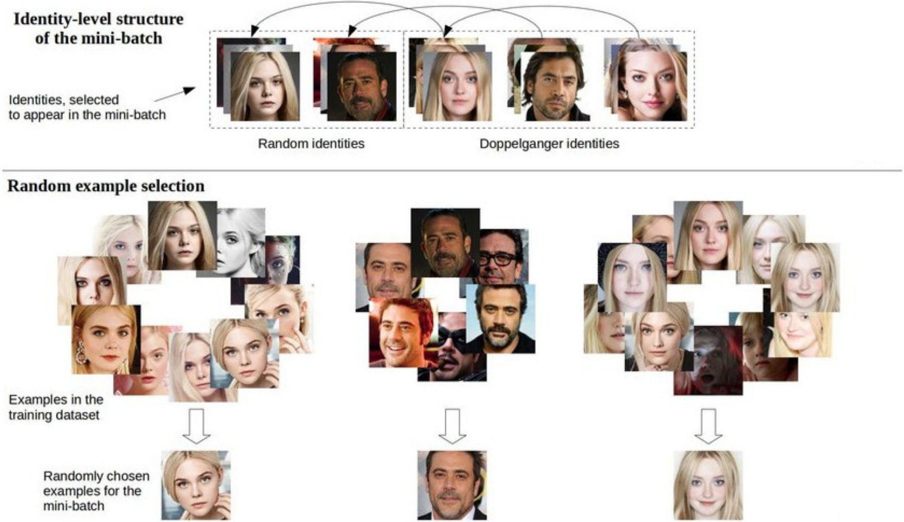
Hard Positive/Negative Mining

Adversarial training

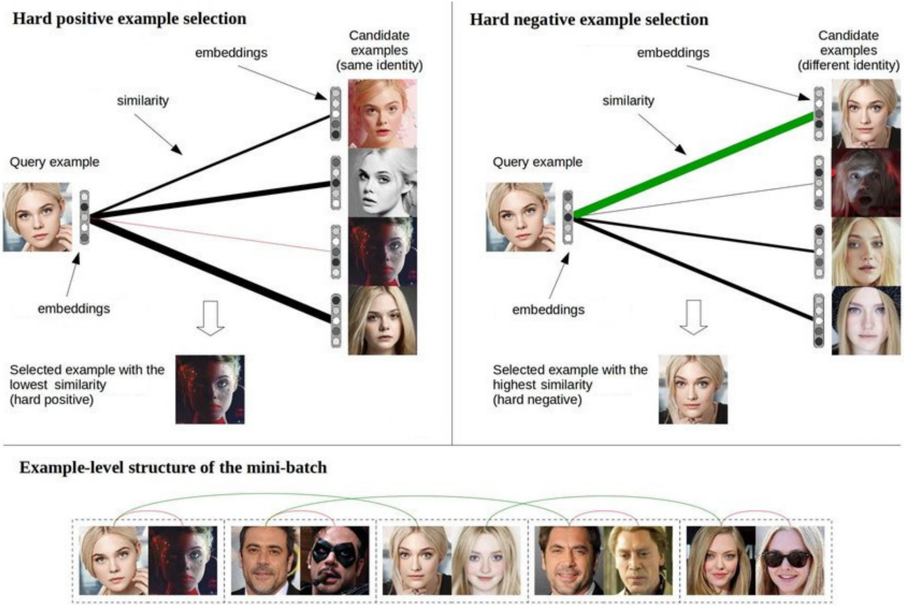
- Train the system from randomly formed mini-batches with balanced positive/negative examples
- Identify hard examples (close to decision boundary) during validation
- Form a new mini-batch by including the hard examples from the the previous iteration.



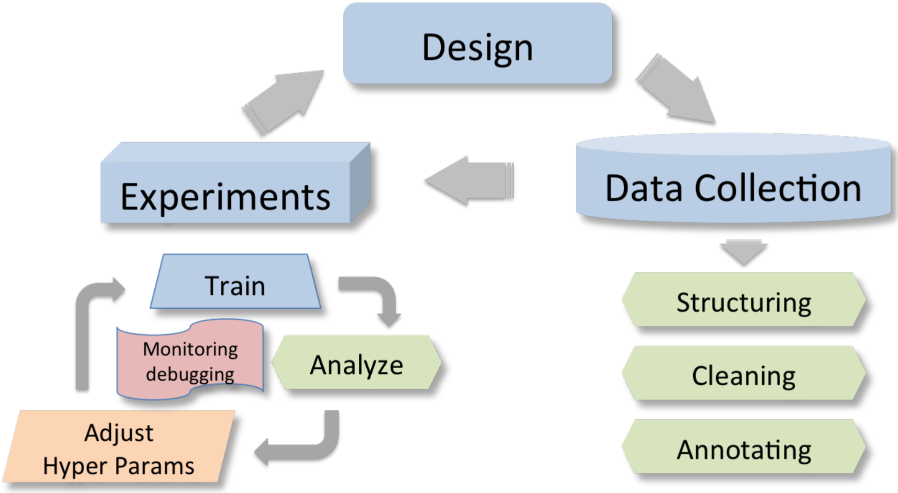
Hard Positive/Negative Mining



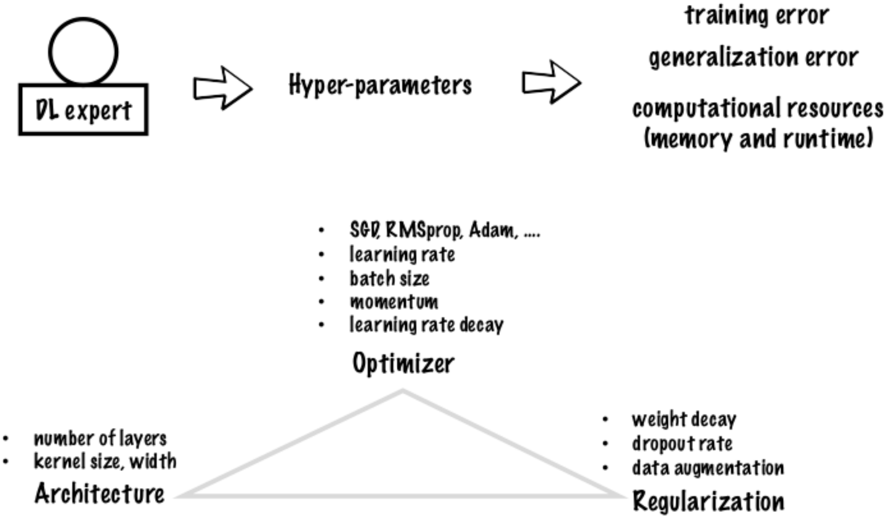
Hard Positive/Negative Mining



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

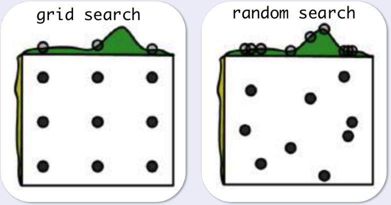
Good rule of thumb

Select the 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)



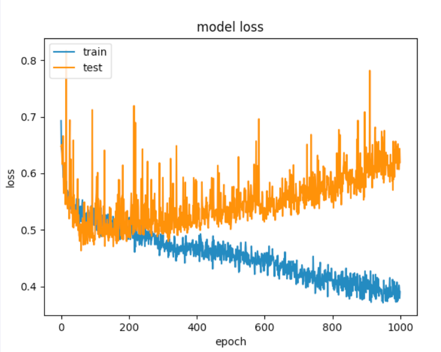
Monitoring and debugging

- 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

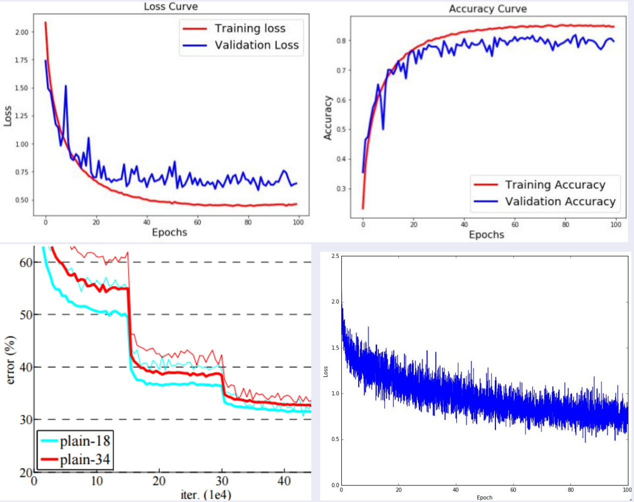
- Observe training error/plot
  - The model should overfit
- Change the optimisation parameters if the model does not overfit
  - If unsuccessful, inspect the code for potential bugs
- Reduce the size of the model (architecture) if model compilation time is long



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

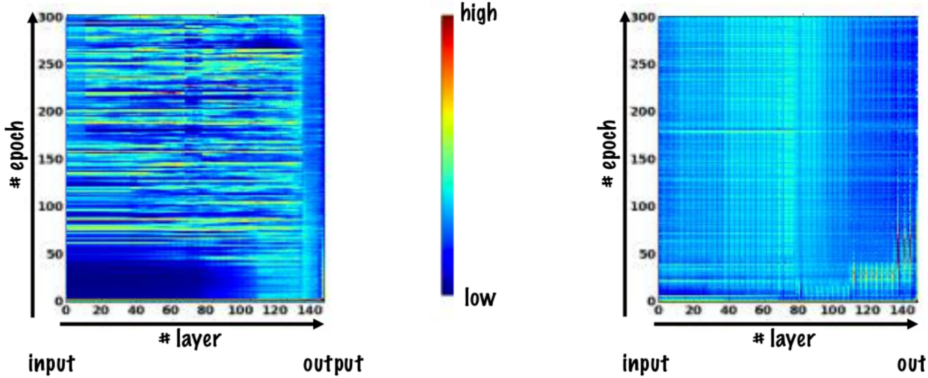
Understanding the underlying causes of the errors

- 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

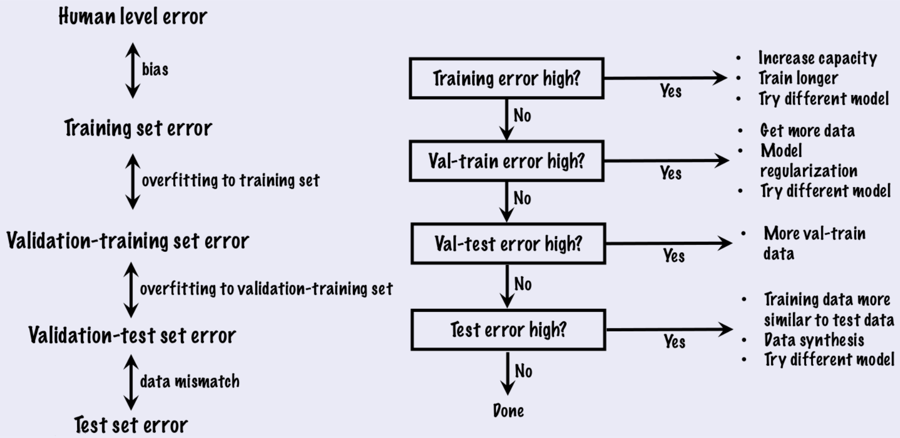
Generalisation problem

- 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