Hochschule Hof
University of Applied Sciences

Efficient learning

Applied Deep Learning

Content

- Faster convergence
  - LR schedulers
  - Regularization
- Accelerators
  - GPU
  - Apple Metall / Neural Engines
  - TPUs
  - Data / Modell parallel
- Zero Shot learning (prompting)
- Modell Destillation

Learning rate schedulers

Hochschule Hof
University of Applied Sciences

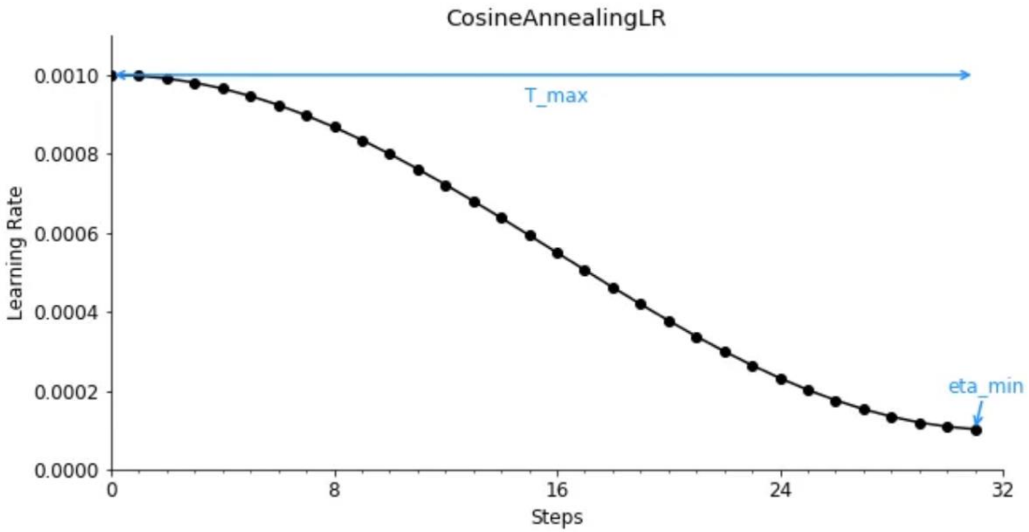
- Optimizer schedulers dynamically adjust the learning rate during training.
- Aim: improve convergence, escape local minima, and ensure generalization.
- Note: Schedulers can be combined

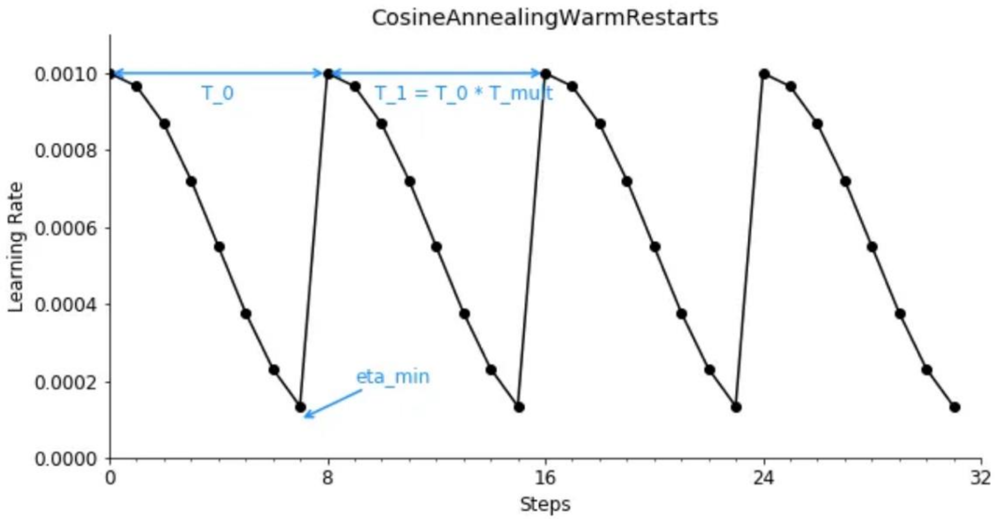
- **Step Decay:** Reduce learning rate by a factor every *n* epochs.

$$\eta_t = \eta_0 \cdot \gamma^{\lfloor \frac{t}{n} \rfloor}$$

- **Cosine Annealing:** Smooth decay using cosine function.
  Suitable for large-scale training.
- **Cyclic Learning Rates (CLR):** Vary learning rate within a range.
  Encourages exploration of the loss surface.
- **Warm-up Schedulers:** Start with small learning rate, increase gradually.

Learning rate schedulers



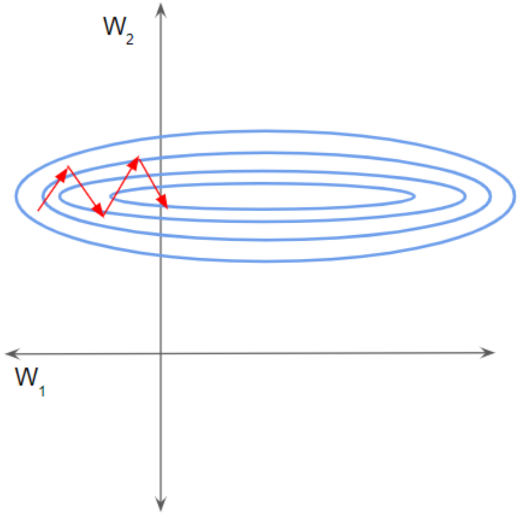


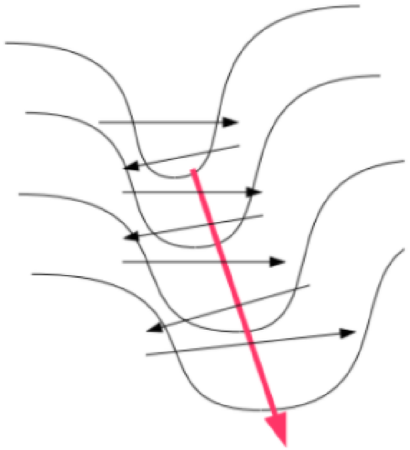
Pytorch Lightning

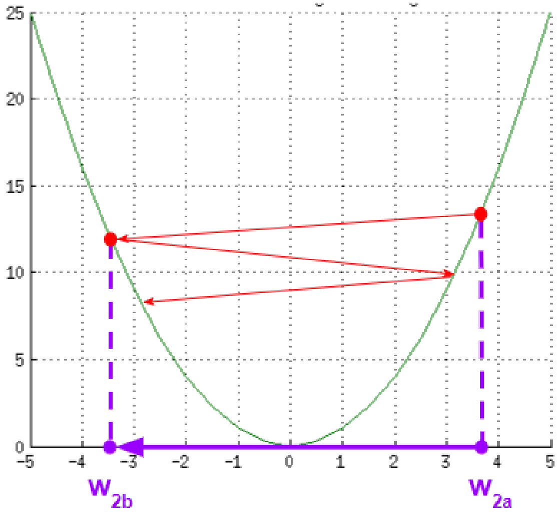
```
def configure_optimizers(self):
    optimizer = Adam(self.parameters(), lr=1e-3)
    scheduler = ReduceLROnPlateau(optimizer,...)
    return [optimizer], [scheduler]
```

Normalization

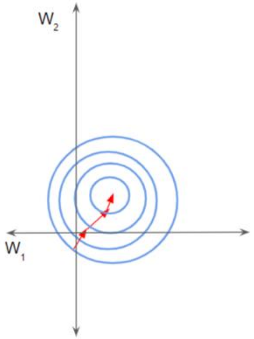
- Problem: unevenly scaled data can lead to problematic optimizer behaviour





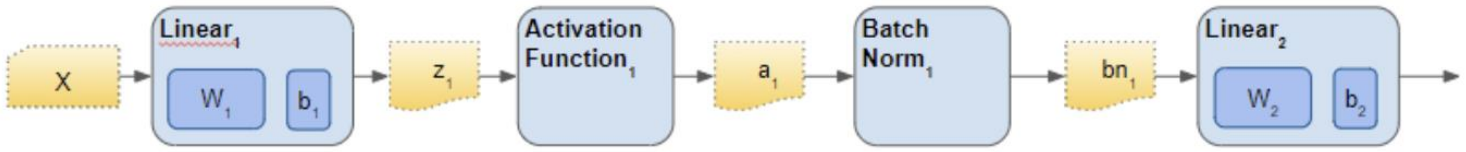


- Goal:



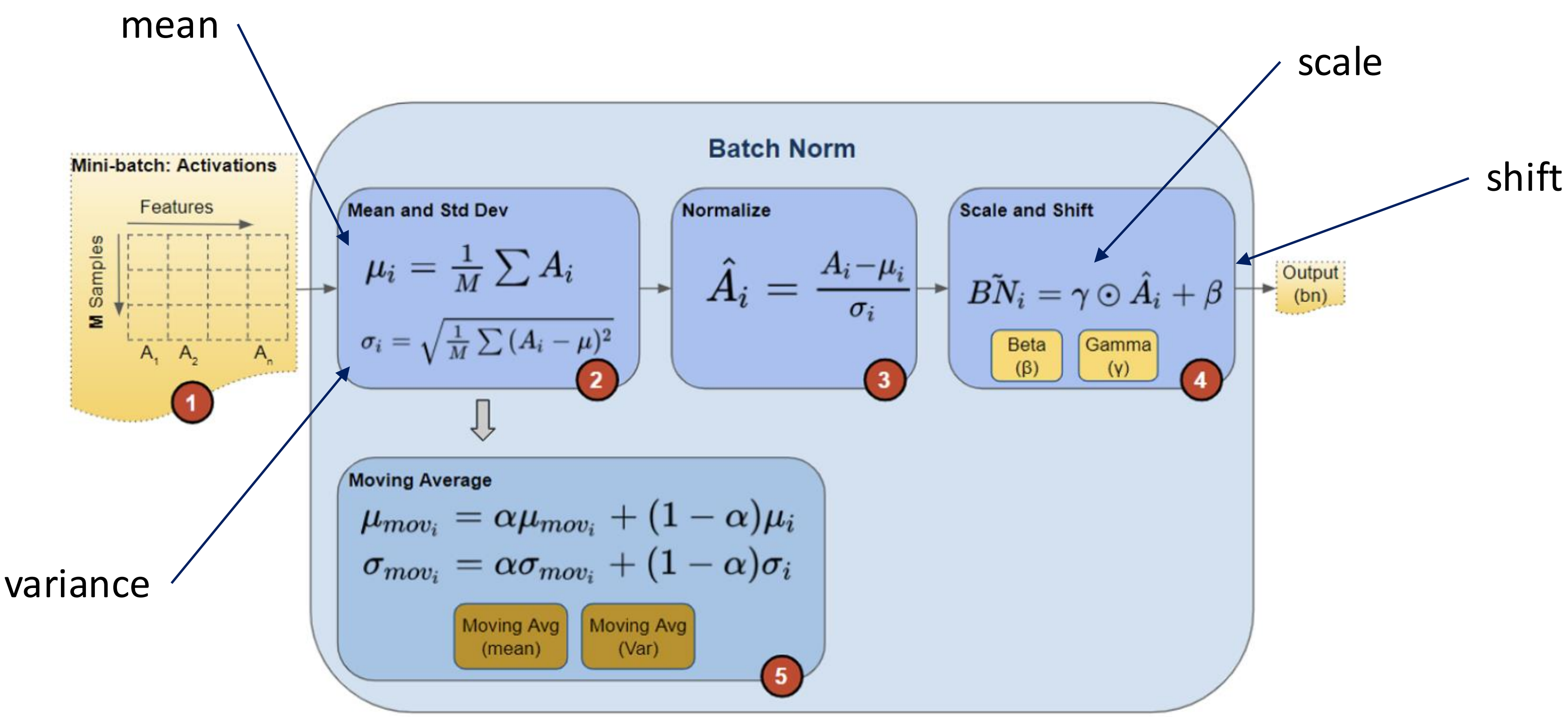
Batch normalization

- Batch norm is another layer inserted into network arch
- Normalizes output before passing to next layer



- 2 learnable parameters mmu & sigma
- 2 stored moving averages values
- Usually places after activation function (original paper: before)

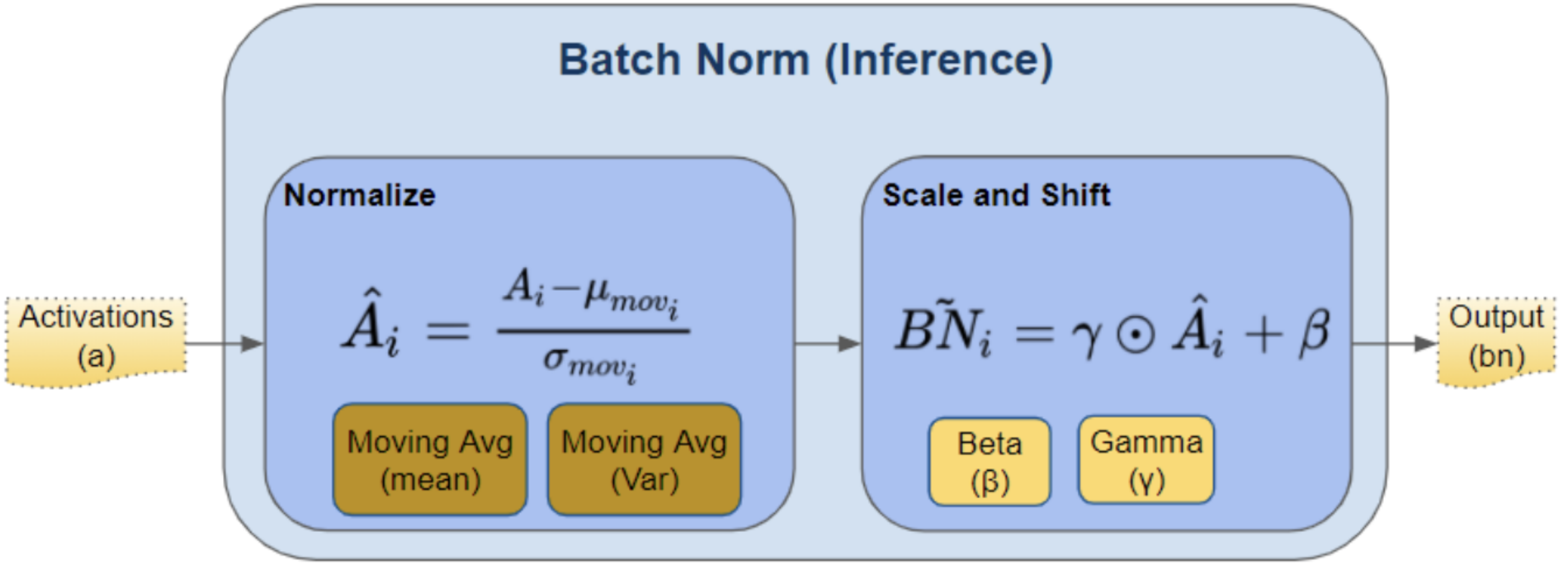
Batch normalization - training



Batch normalization - inference

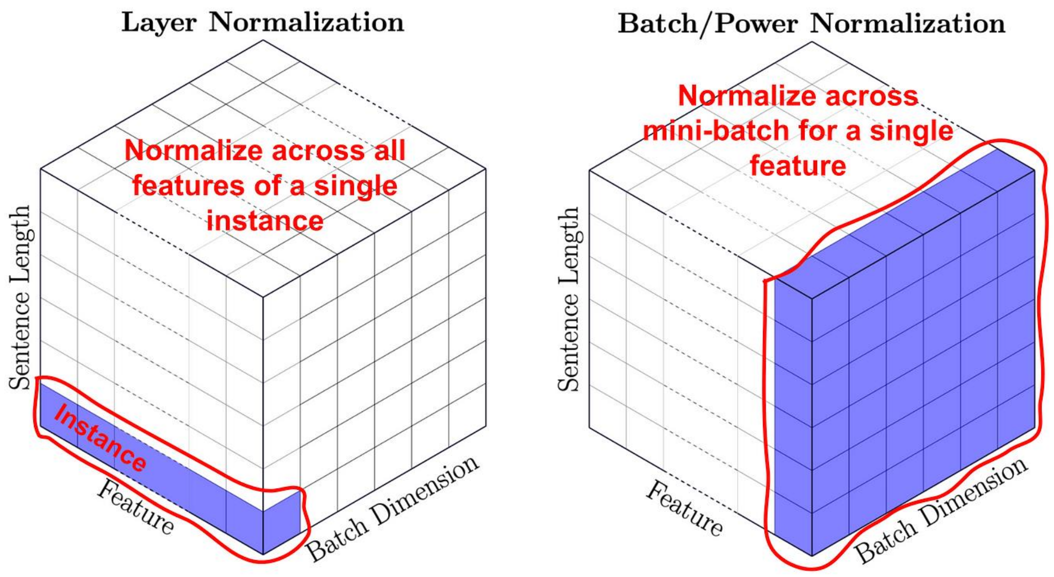
Hochschule Hof
University of Applied Sciences

- During inference, we only have a single sample, not a mini batch → mean and variance?
- Moving averages used as good proxies



Layer normalization

- Calculates the mean and variance of the activations for each individual input sample, considering all features
- BN
  - Good where features are related, like computer vision (fully connected, CNNs)
- LN
  - Reduces internal covariance shift
  - Good with RNNs
  - Same operations in train and inference



Regularization

- Shall prevent overfitting

L1 regularization

- add penalty term to loss which depends on weights
- Scaling factor alpha

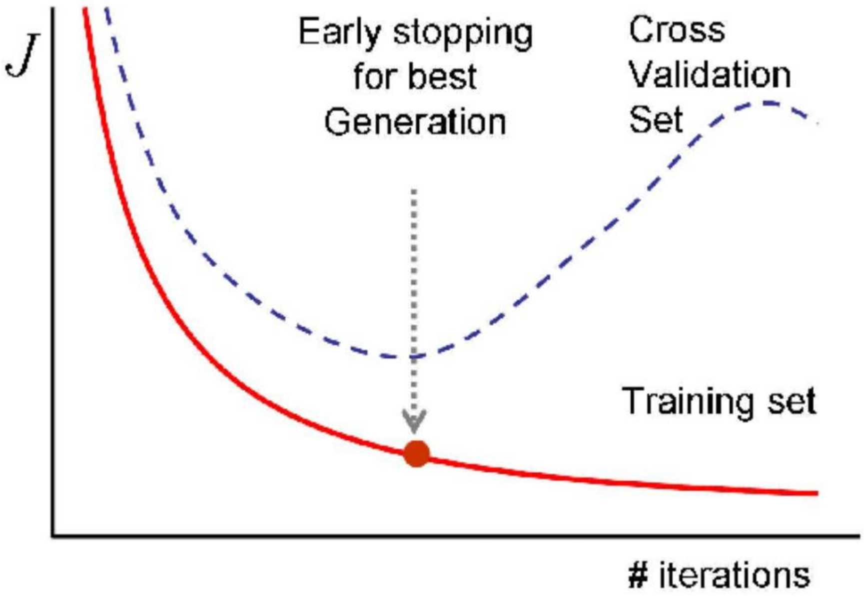
$$Loss = Error(y, \hat{y}) + \alpha \sum_{i=1}^{N} |w_i|$$

$$w_{\text{new}} = w - (\lambda \frac{\partial \text{Error}}{\partial w} + \lambda \frac{\partial(\alpha \sum_{i=1}^{N} |w_i|)}{\partial w})$$

$$\frac{\partial(\alpha \sum_{i=1}^{N} |w_i|)}{\partial w} = \begin{cases} \alpha & w > 0 \\ -\alpha & w < 0 \end{cases}$$

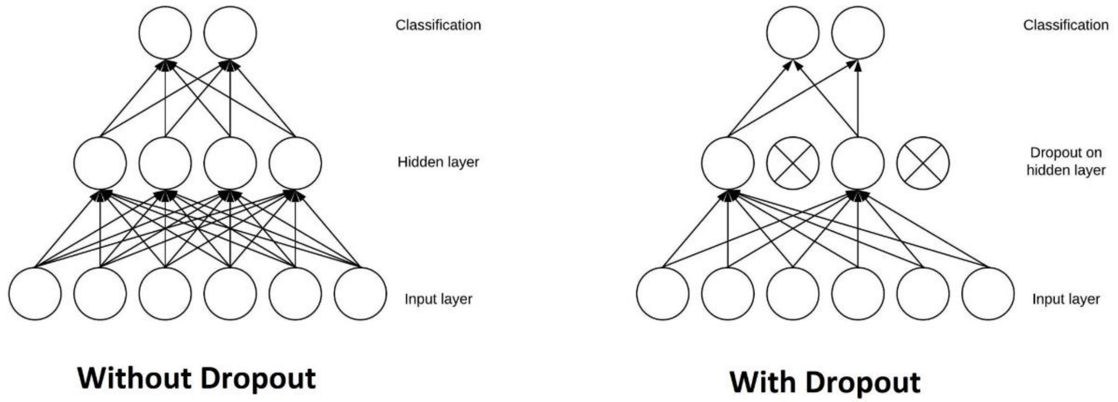
Regularization

Early stopping



Dropout

- Randomly dropping neurons in a layer (no update on backprop)
- Creates redundancies in the network, reduces complexity



Regularization

Label smoothing

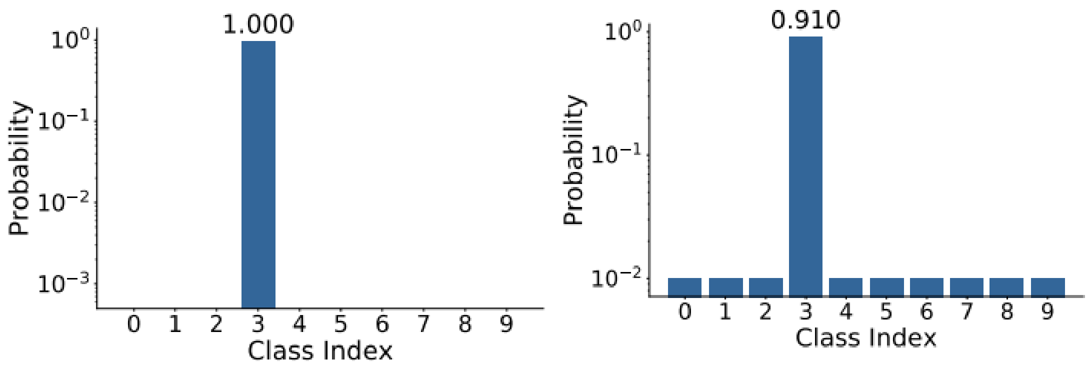
- Replaces one-hot labels with a weighted distribution
- A technique that prevents the model from becoming too confident.
- Reduces overfitting and sharp decision boundaries.

Formula:

$$y_{\text{smooth}} = (1 - \epsilon) \cdot y_{\text{one-hot}} + \frac{\epsilon}{K}$$

where ϵ = smoothing factor, K = number of classes

Common setting: $\epsilon = 0.1$



(a) Hard Label (b) LS

Accelerators

Hochschule Hof University of Applied Sciences

- Training NN is compute intensive
- Accelerators reduce training time and energy consumption.
- Key concepts
  - Speed up matrix operations (e.g., convolutions, multiplications)
  - Enable large-batch parallel training
  - Efficient memory handling (bandwidth, latency)

Accelerators

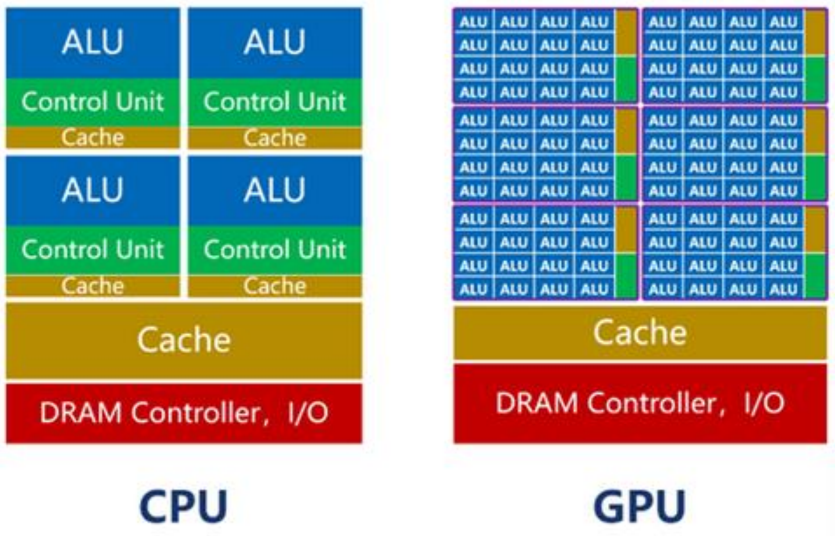
Hochschule Hof
University of Applied Sciences

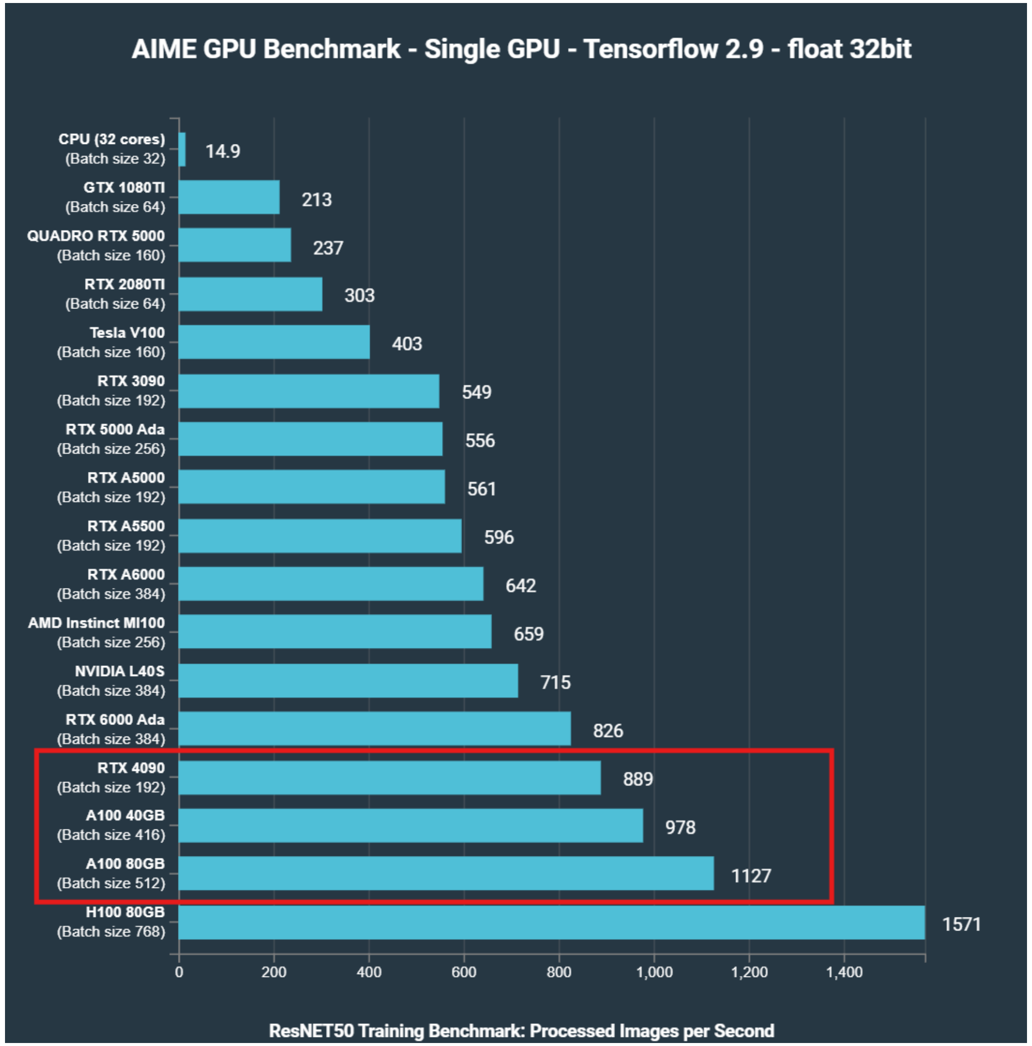
https://www.youtube.com/watch?v=ZrJeYFxpUyQ

Cuda

Hochschule Hof
University of Applied Sciences

- Compute Unified Device Architecture
- Parallel computing platform by NVIDIA.
- Deep integration with PyTorch, TensorFlow, JAX.





Apple

Apple Metal

- Low-level GPU API optimized for macOS and iOS.
- Leverages Apple's Neural Engine and M-series GPUs.
- Pytorch support over Core ML Tools
- Offers parallel computing ability for apple GPUs, but fewer control than CUDA

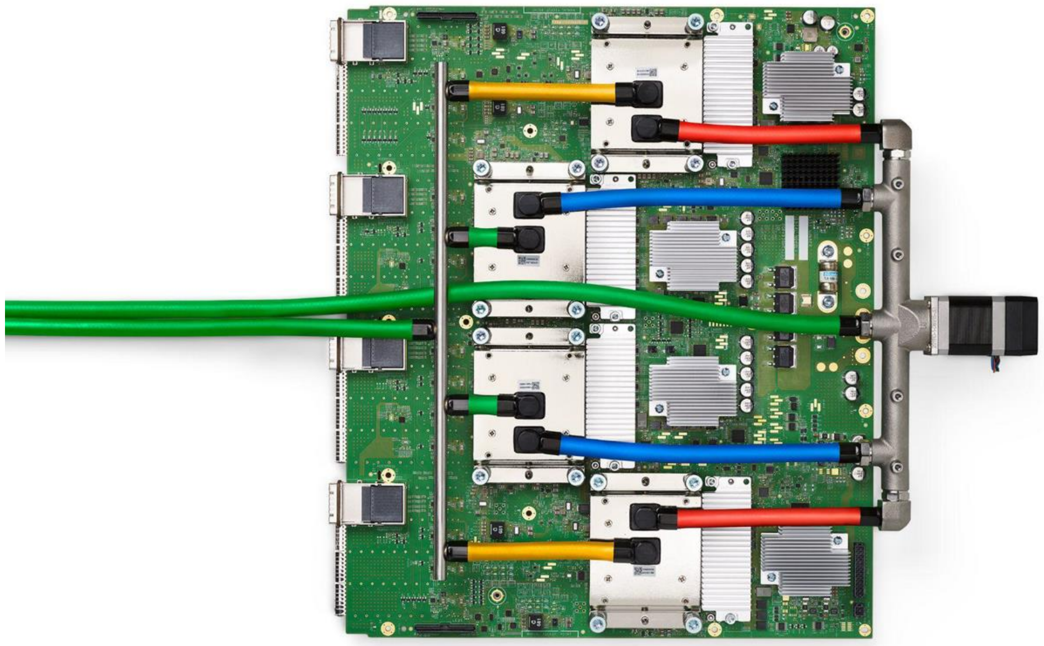
Apple Neural Engine

- Dedicated accelerator since A11 Bionic
- Optimized for inference
- Supports up to 15 Tflops
- Not used for training
- Over Apple Core ML

Google TPU

Hochschule Hof
University of Applied Sciences

- Developed by Google for neural network trainings
- Designed for Tensorflow applications
- For high volume, low precision
- Optimized for input & output per Joule!
- Best for CNN-like

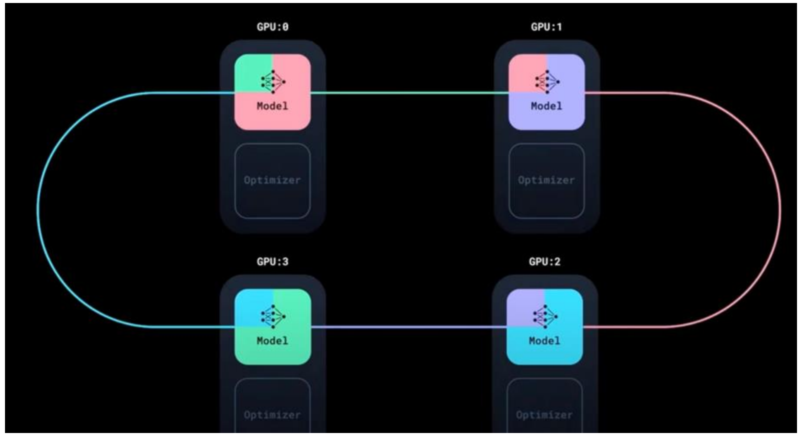


4 ASICs with cooling, Gen4 TPU

Parallelism

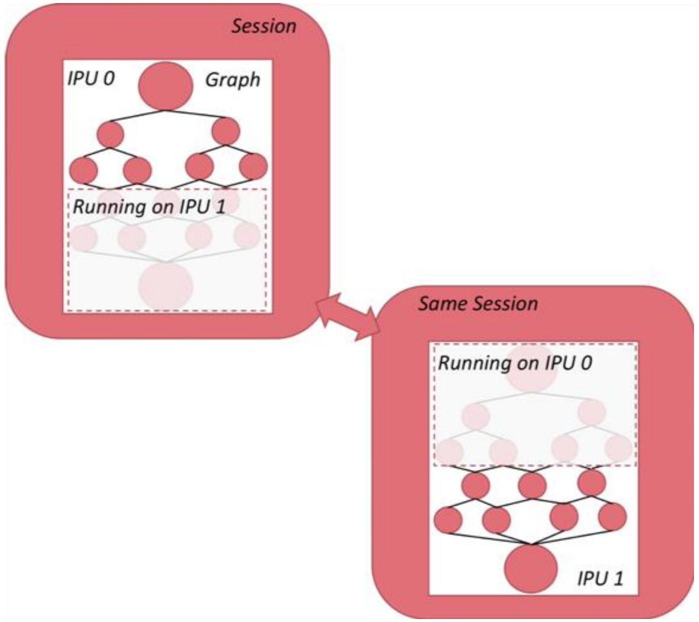
Data parallel (DDP)

- Same model is stored in different computing nodes
- Data is split onto nodes
- After backward step, gradients are accumulated and distributed to every node
- Grad exchange is done in ring – bucket → nodes are always busy / no waiting



Model parallel

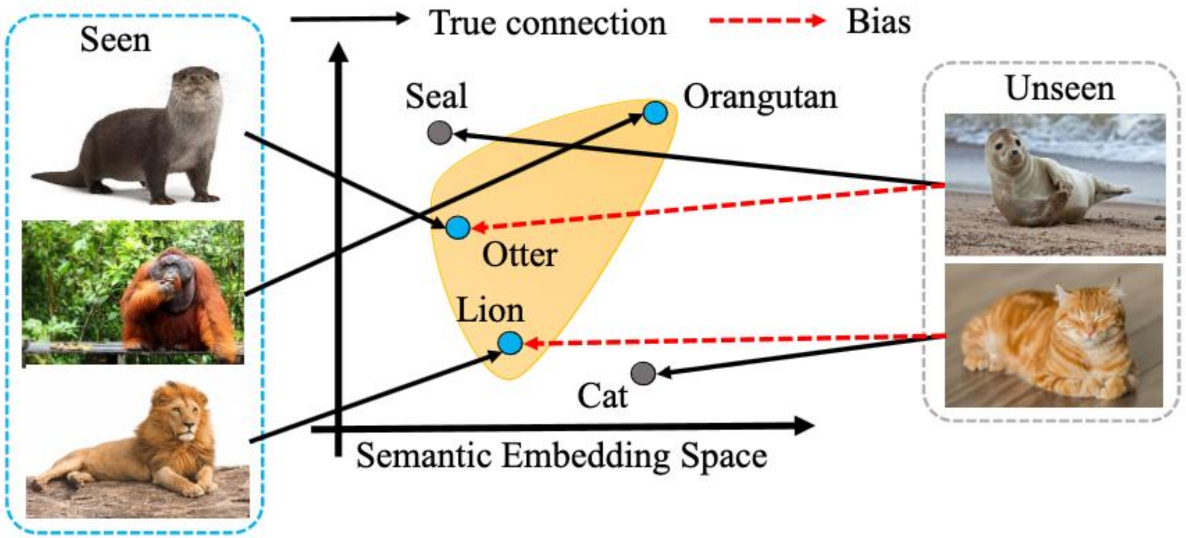
- Split model onto different nodes
- Model is sharded by layers
- Sync bottleneck



Zero shot learning

Images

- Model extracts meaningful features from data
- Organizes in internal embedding space
- Uses embedding space to identify new unseen data
- Embedding arithmetics can apply



LLMs

- Leverage internal abilities without additional training
- Usually in-context learning
- Even possible without in-context training, only using prompts

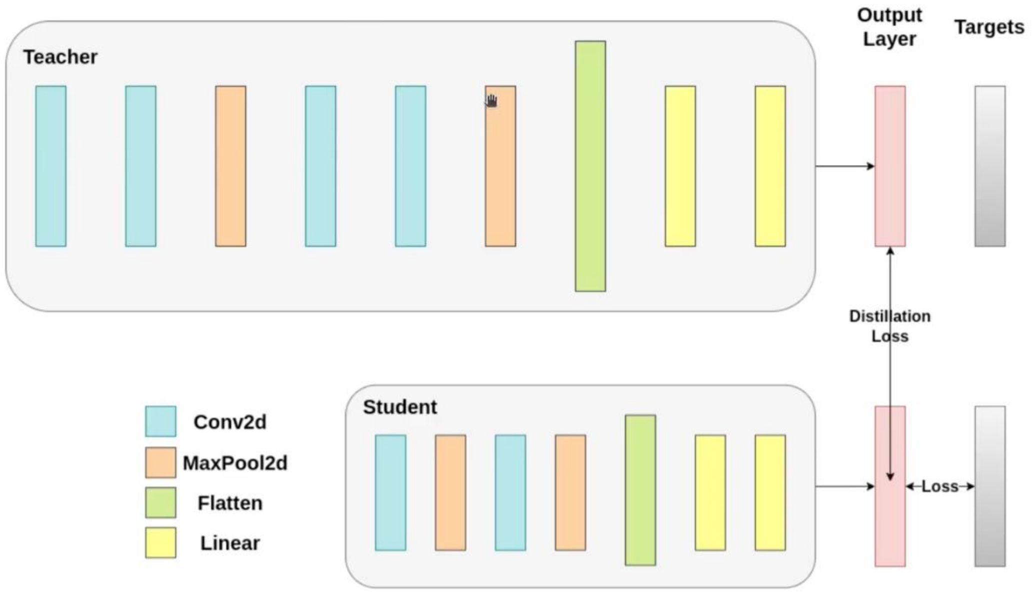
```
Classify the text into neutral, negative or positive.
Text: I think the vacation is okay.
Sentiment:
```

Output:

```
Neutral
```

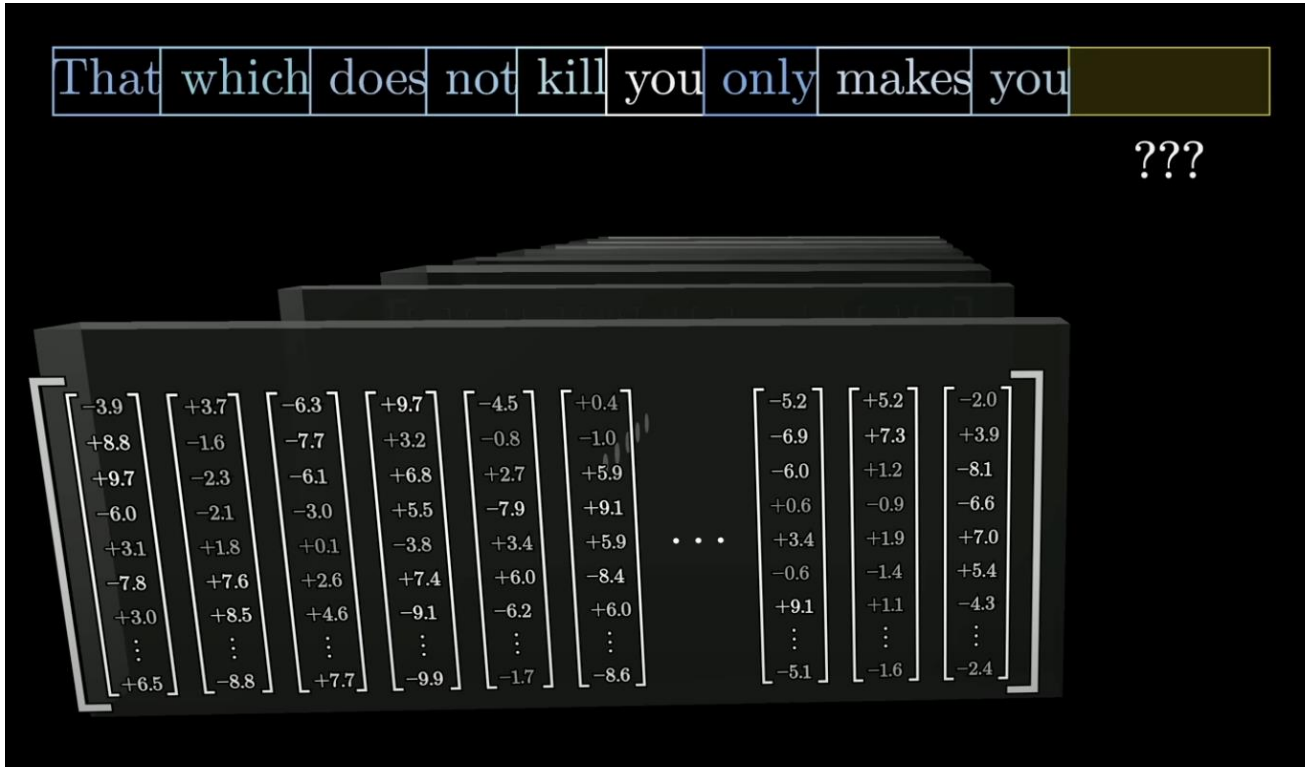
Model destillation

- Also knowledge destillation
- Abilities of larger model (teacher) are conveyed to a smaller model (pupil)
- Implementations
  - Using GANs to make pupil indestinguishable from teacher
  - Cross-Modal destillation: one works in images, one in text
  - Soft target dest.: using logits instead of hard outputs



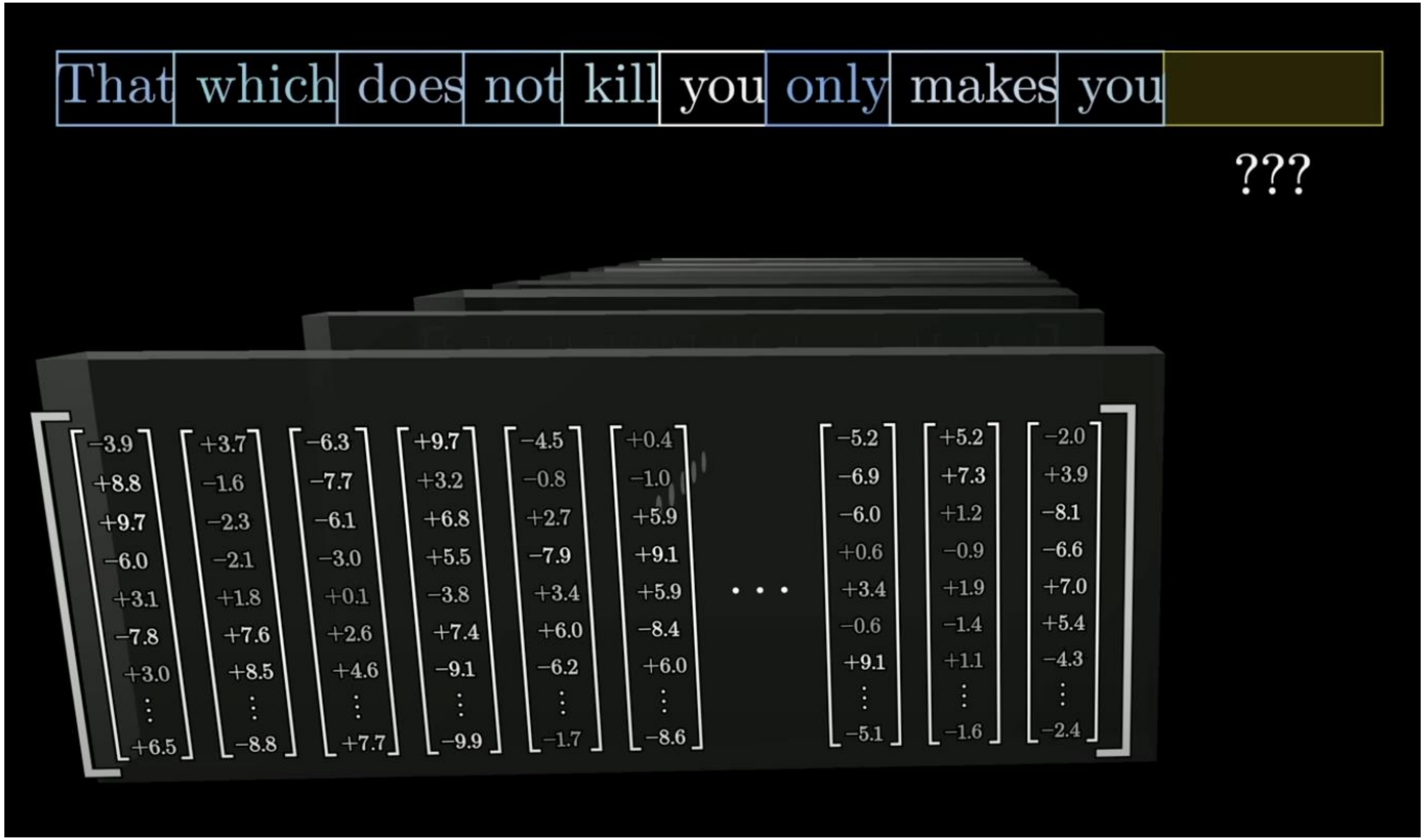
Model internals

- Word are stored as embeddings
- Size of embedding around twelve thousand entries (GPT 3)



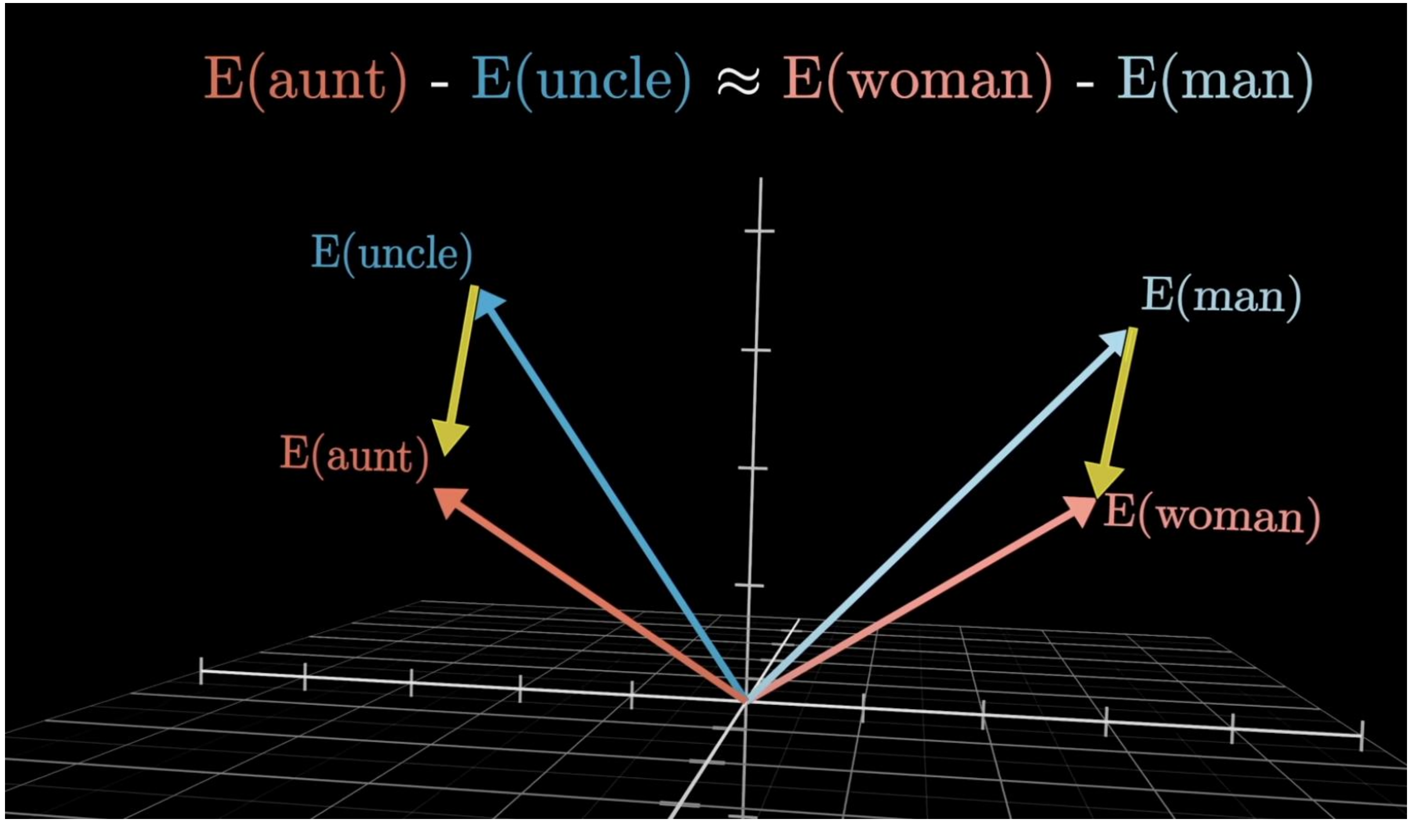
Model internals

- Word are stored as embeddings



Model internals

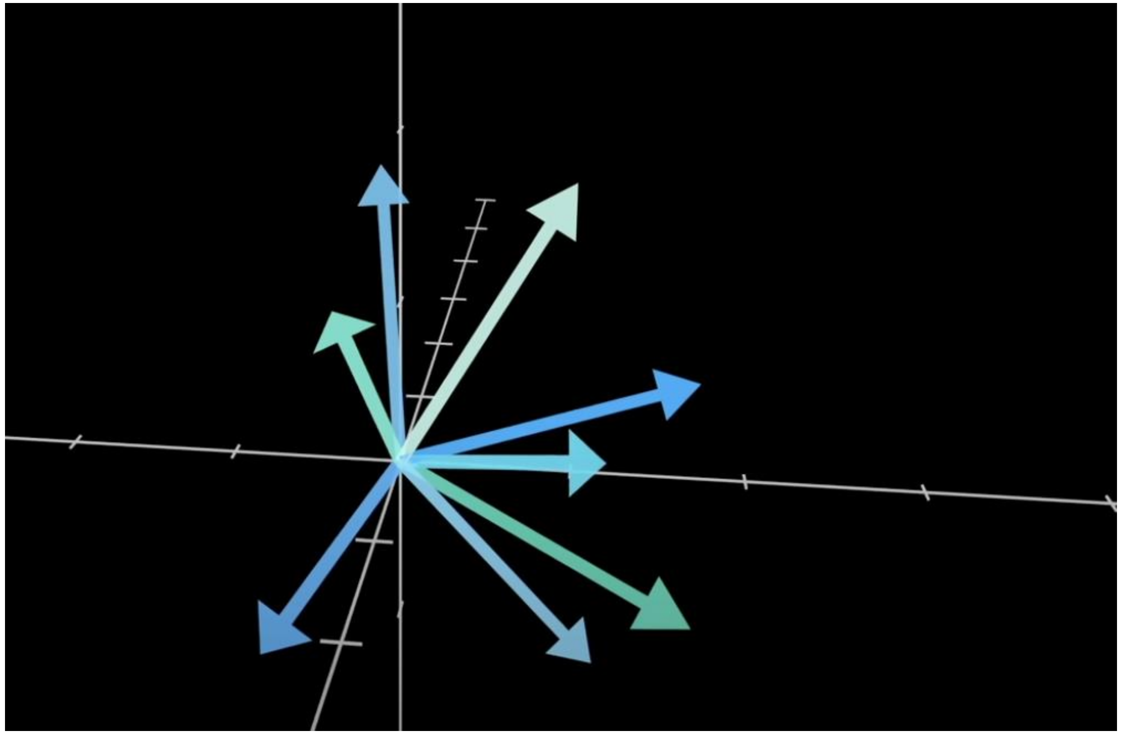
- Embeddings can be found in each layer
- Each perpendicular direction can be associated with a concept



Model internals

Hochschule Hof
University of Applied Sciences

- Problem: can we only store twelve thousand concepts?
- Yes!, but they are not completely perpendicular anymore (superposition)
- For 100 dimensions, we can find around 10.000 vectors
- Amount grows exponentially!



Wrap up

- Faster convergence by scheduling and regularization
- Accelerators
  - Specific hardware
  - Data / Modell parallel
- Modell Destillation