Essential Techniques for Deep Learning to Avoid Overfitting

Harsh Kumar, Gojek

13th July 2024

Why is difficult to train neural networks

A Recipe for Training Neural Networks, *Blog by Andrej Karpathy (link)*

1. Neural net training is a leaky abstraction
   - Not plug-and-play like standard software APIs

2. Neural network training fails silently
   - Beyond syntactic errors, error surface is large

The Recipe

Understand Data → Get Dumb baseline → Overfit → Regularize → Tune → Squeeze out the juice

What is Overfitting?

- When model learns the training data too well, including noise and outliers, resulting in poor performance on new, unseen data
- The model captures details that are specific to the training data but does not generalize to the broader datasets

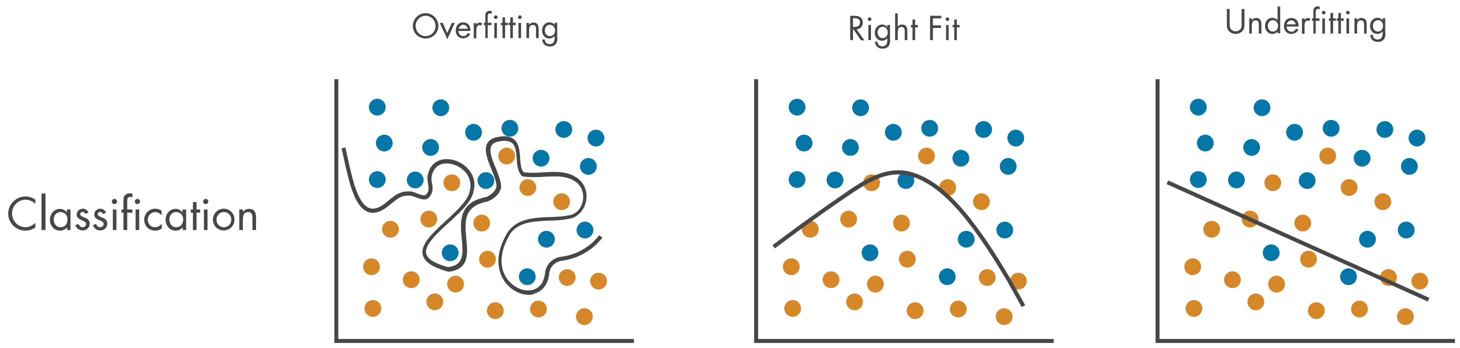


Image Source: https://www.mathworks.com/discovery/overfitting.html

We want good prediction accuracy not only on the training data, but also on unseen real-world data

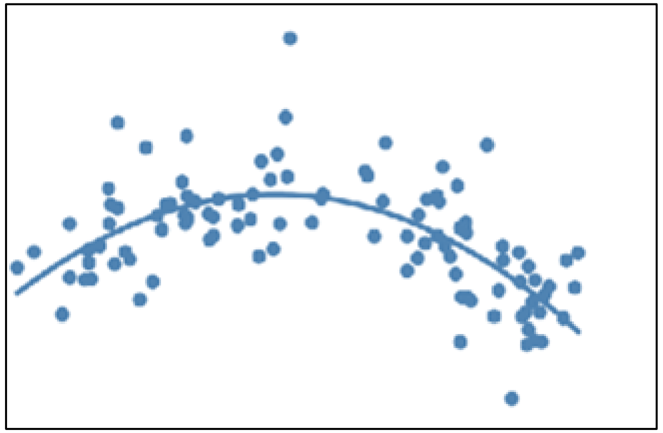
What causes overfitting?

1. More complex model than required
2. Insufficient training data
3. Too many irrelevant features

<u>Using very complex model with low amount of training data</u>

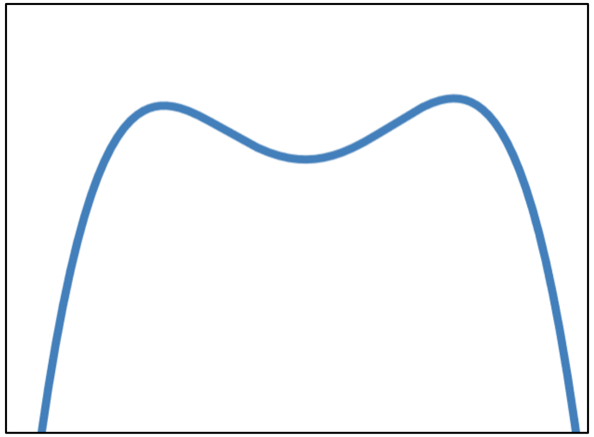
Data has quadratic relationship in reality

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$



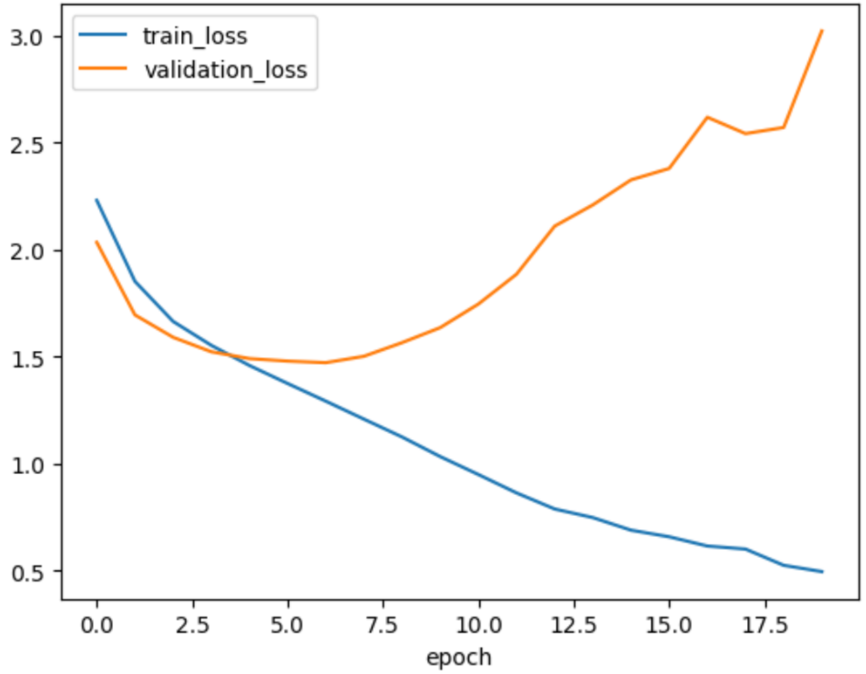
Model equation is complex

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$



How to detect overfitting?

1. Training loss much lower compared to validation loss
2. Validation loss increases as number of epochs[1] increase



Always track model performance on properly constructed validation dataset

1. An epoch is when all the training data is used at once and is defined as the total number of iterations of all the training data in one cycle for training the machine learning model

Techniques to Avoid Overfitting

First things to check

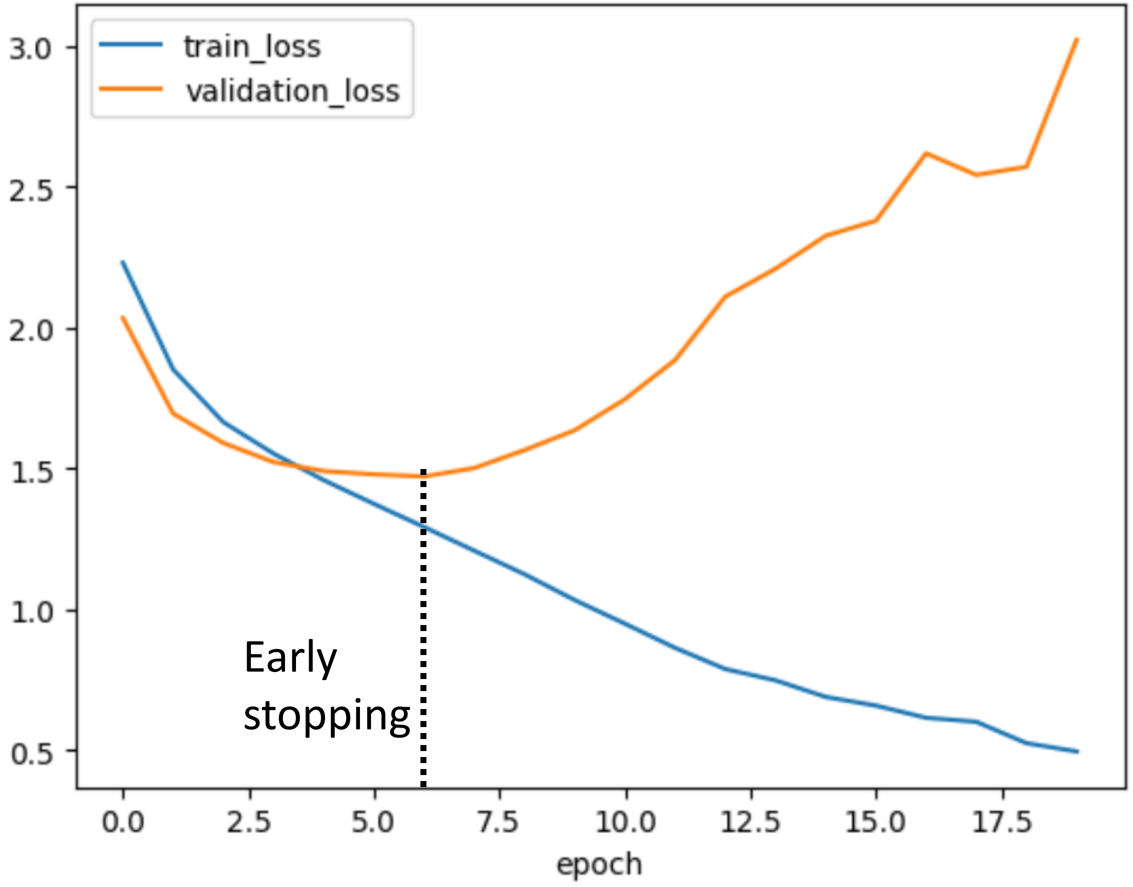
Overfitting happens when the model is too complex compared to the patterns in the input data

1. **Understand your data**
    - Is the training and validation/test data coming from the same distribution?
    - Is the data clean enough? Check for outliers, missing values, duplicates, corrupted images or wrong labels
    - Are the features normalized? Is there a class imbalance?
    - Can we get more data – easily, quickly, cheaply?

2. **Choose the right model**
    - Reduce model parameters (# hidden layers, # neurons per layer)
    - Try simpler approaches like random forest, logistic regression, CART

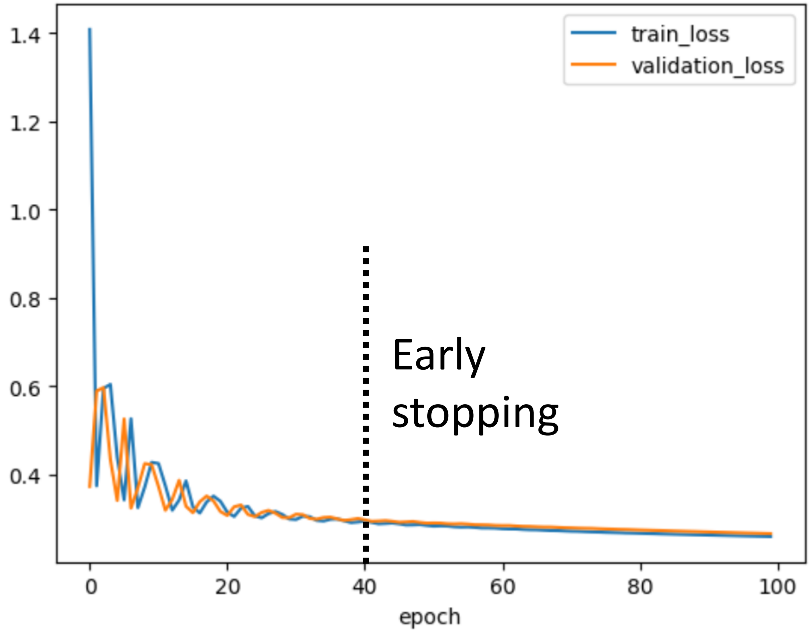
Early Stopping

Stop training when validation loss stops reducing



Early Stopping

Use early stopping even validation loss is not increasing



Why: Reduces chances of overfitting on unseen data + Saves compute and time

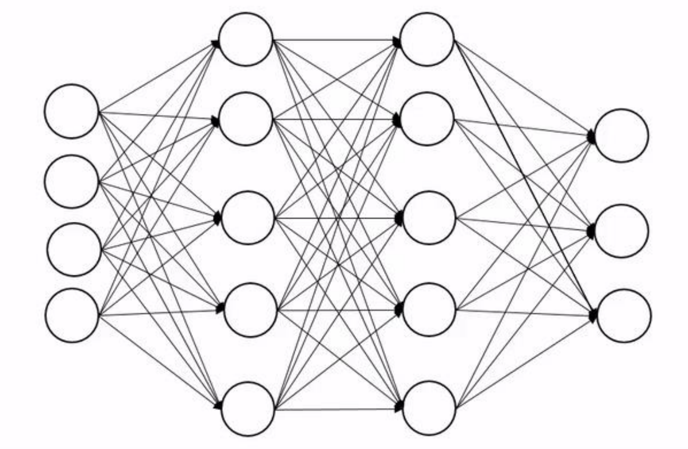
How: Stop training when validation loss doesn't reduce by more than x%

Dropout

What: Dropout involves randomly ignoring a fraction of neurons during training

How:

1. Dropout probability determines the fraction of neurons ignored
2. Remaining neurons are scaled up (if dropout probability if 50%, then the output of remaining ones are multiplied by 2)
3. No dropout during inference time



Source: https://www.analyticsvidhya.com/blog/2022/08/dropout-regularization-in-deep-learning/

Why:

- Reduces co-adaptation of neurons
- Acts as an ensemble of networks
- Promotes redundant representations

Dropout probability is typically 20% to 50%, but experiment for your use case

L1 and L2 Regularization

Add a penalty to the loss function to prevent parameters of the neural network from becoming too large

L1 Regularization (Lasso): $Regularized\ Loss = Loss + \lambda \sum |w_i|$

L2 Regularization (Ridge): $Regularized\ Loss = Loss + \lambda \sum w_i^2$

where λ is the regularization parameter, w_i are the model parameters and n is the number of parameters

Difference between L1 and L2

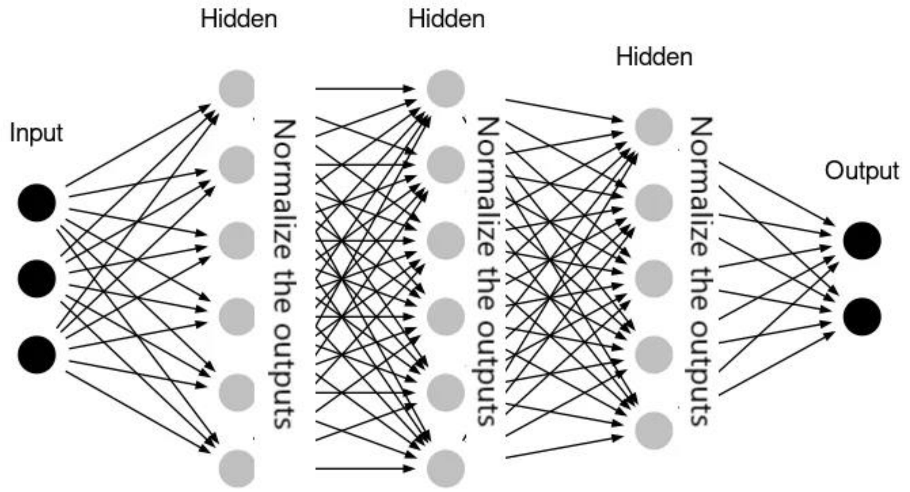
- L1: Sparsity - Good for feature selection, can introduce bias
- L2: Weight decay - Good for stability, does not select features

Batch Normalization

Normalization: Rescaling **input features** to make the values of all features lie in a common range

- Min-Max Normalization $\left(\frac{X - X_{min}}{X_{max} - X_{min}}\right)$
- Standardization using mean and standard deviation $\left(\frac{X - \mu}{\sigma}\right)$

Batch Normalization: Normalizing outputs of each neuron across mini-batches of input data



Source: https://medium.com/@abheerchrome/batch-normalization-explained-1e78f7eb1e8a

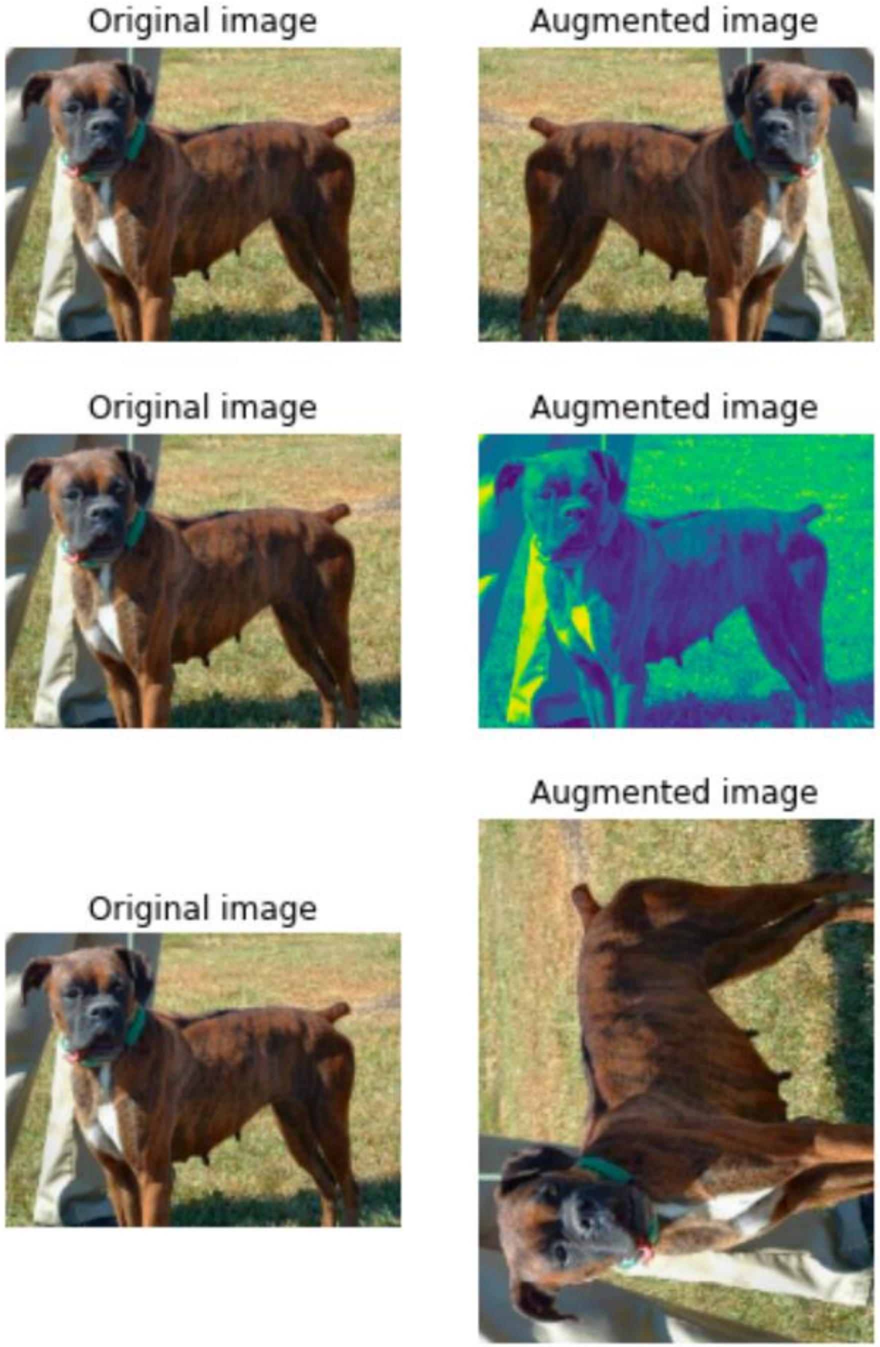
Without batch normalization, output of one of the neuron can become quite large which can propagate throughout the network

Data Augmentation

Increase the diversity of training dataset without collecting new data by creating modified versions of existing data

Techniques for augmenting image data:

1. **Geometric Transformations**: Rotate, Zoom in or out, Flip, Crop
2. **Color & Lighting Adjustments**: Brightness, Contrast, Saturation, Hue
3. **Noise Injection**: Add random noise
4. **Synthetic Data Generation**: Using GANs (Generative Adversarial Networks)



Source: https://www.datacamp.com/tutorial/complete-guide-data-augmentation

Data Augmentation

Techniques for augmenting audio data:

1. Pitch shifting
2. Speedup/Slowdown
3. Adding background noise

Techniques for augmenting textual data:

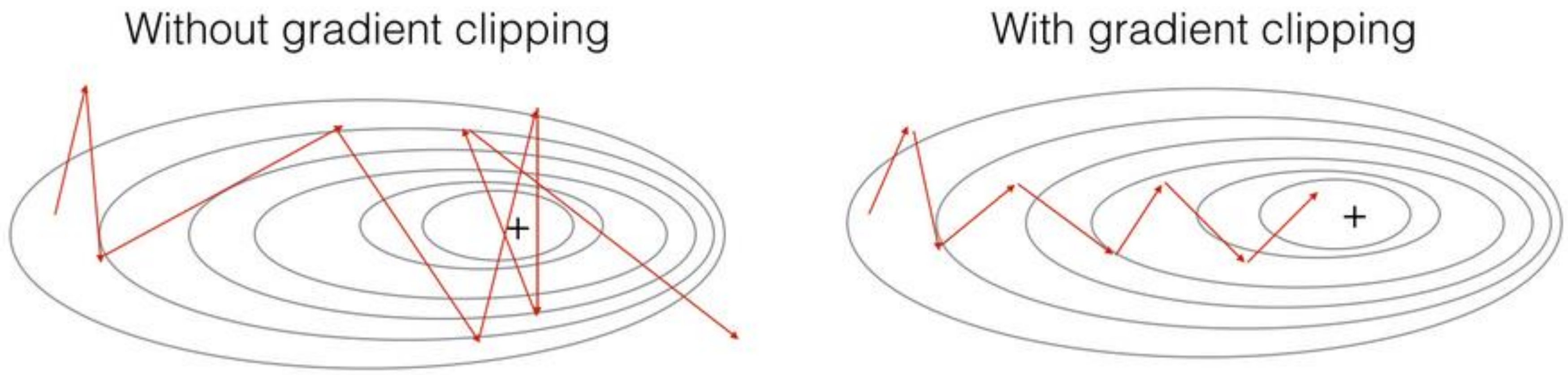
1. Synonyms
2. Translations
3. Adding random words for noise

Benefits

1. Increased Dataset Size
2. Improved Generalization
3. Understanding of Model Limitations

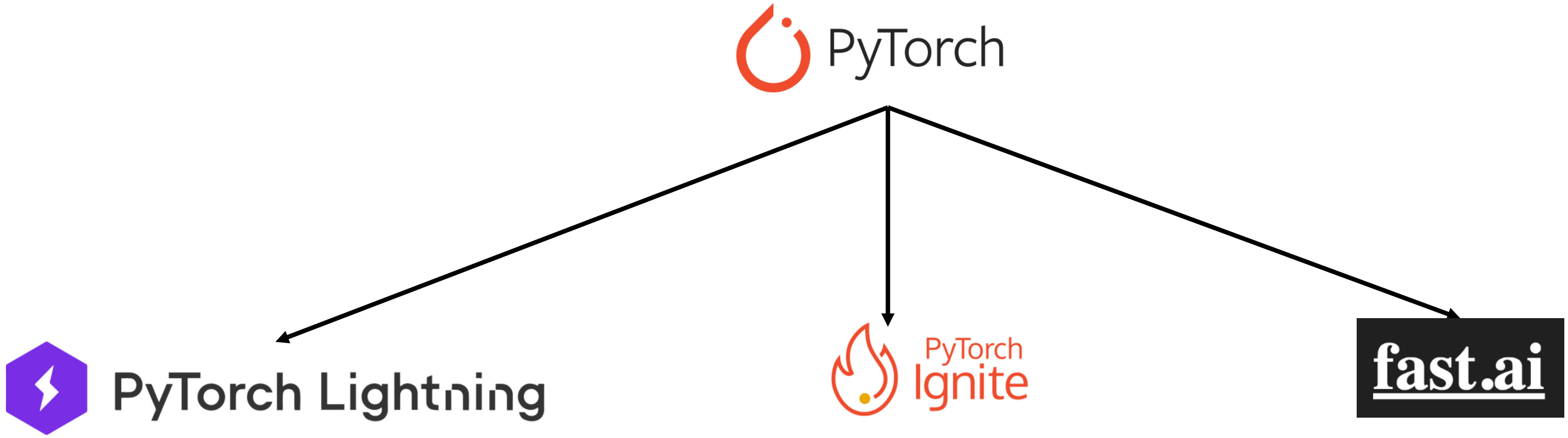
Gradient Clipping

- Generally used to **prevent exploding gradients**
- We clip the gradients to a maximum value to ensures that the gradients are scaled down to a reasonable size
- Gradient clipping also acts as **a form of regularization** which improves the generalization of the model



Source: https://deepai.org/machine-learning-glossary-and-terms/gradient-clipping

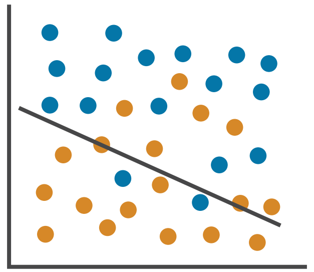
Combining all together with Lightening



High level frameworks for PyTorch for abstracting out boilerplate code

Finally, be carful not to underfit

Underfitting



Model is too simple to capture the underlying patterns in the data

Why fix underfitting?

Our effort is to train a good model with performs well on real-world data. Underfitted model doesn’t learn patterns even on training data, resulting in poor real-world performance.

Improper use of regularization techniques can results in underfitting

Don’t underfit in an effort to mitigate overfitting

Fixes

1. Increase model complexity
2. Feature engineering
3. Fix regularization parameters
4. Longer training duration

Techniques for further exploration

1. **Stochastic Depth** (Paper)**:** Some layers are randomly dropped during training, but no dropping during testing

2. **DropConnect:** Generalized version of Dropout where we disable individual weights instead of neurons, so a neurons can remain partially active

3. **AutoAugment/RandAugment** (Paper)**:** Automatically searches for the best data augmentation policies

4. **Shake-Shake regularization** (Paper)**:** When building a deep neural network, instead of adding the outputs of each branch together, combines them in a random way during training to introduce noise

5. **Mixout** (Paper): A regularization technique similar to dropout used during transfer learning. It replaces a fraction of the model's parameters with the corresponding parameter of a pre-trained large model.

6. **Masked Language Modeling (MLM):** Used in NLP, where parts of the input are masked and the model learns to predict the masked parts

Thank you!!

Link to presentation and notebooks