Introduction to DNN Training: Fundamentals, Process and Best Practices

Kevin Weekly, Ph.D.

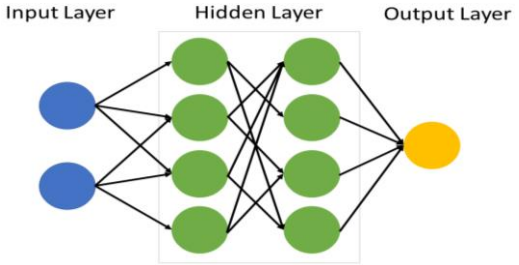
CEO

Think Circuits

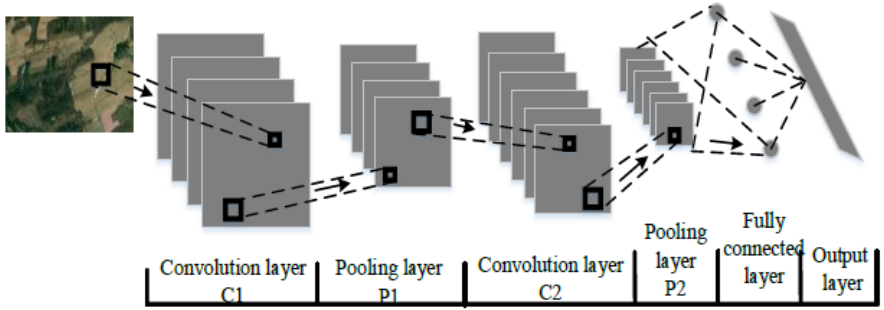
THINK CIRCUITS™

Intro to neural networks

- Neural Network – a bio-inspired computational model consisting of nodes (neurons) connected by mathematical relationships (network) and parameters.
- Deep Neural Network – Incorporates a multitude of hidden layers to learn complex and hierarchical information.
- Convolutional Neural Network – Incorporates convolutional layers well suited for spatially gridded data like images.



A prototypical neural network



Example CNN architecture[1]

[1]Figure from Li, Y., Liu, C., Zhao, W., & Huang, Y. (2020). *Multi-spectral remote sensing images feature coverage classification based on improved convolutional neural network.* Mathematical Biosciences and Engineering, 17(5), 4443–4456. https://doi.org/10.3934/mbe.2020245

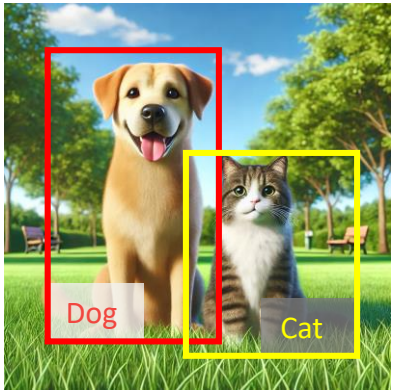
Tasks in computer vision models

- **Classification**: What “class” does this image belong to?
- **Object detection**: What objects/classes are in this image?
- **Segmentation**: What class does each pixel belong to?

Other computer vision tasks:

- **Visual inertial odometry, localization and mapping, image registration :** What is the transform between two images?
- **Vision-language models, end-to-end AI:** What is the best output given this image + context?

Classification



Detection


Segmentation

What does it mean to train a model?

- Trainable parameters

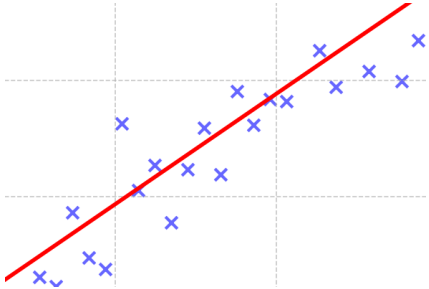
Regression: $\hat{x} = \boldsymbol{a}y + \boldsymbol{b}$

Classification: $x = \begin{cases} 0 \text{ if } \boldsymbol{a}y + \boldsymbol{b} < \boldsymbol{t} \\ 1 \text{ otherwise} \end{cases}$

- Hyperparameters
  - Model Architecture – layers, activation function.
  - Optimization – type, learning rate, etc..
  - Regularization – dropout, early stopping.
  - Training strategy – epochs, loss function.

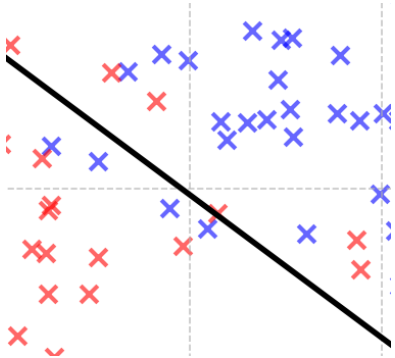
Regression Example

Find best-fitting line



Classification Example

Find line which separates two classes

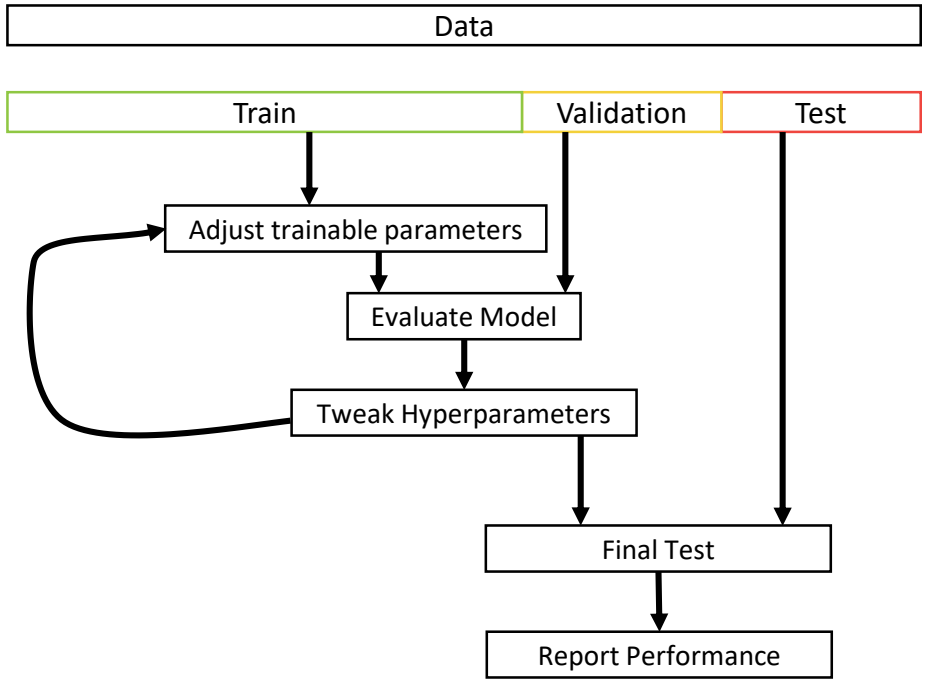


What do you need to train a model?

- Labelled data – A corpus of inputs and their desired outputs. Sources:
  - **Hand labeling** from outside datasets or tools like Labelbox.
  - **Synthetic data** from simulations and data augmentation.
  - **Transfer learning** where labels come from another model.
- Data variation – needs to include broad coverage of the application domain even after splitting.

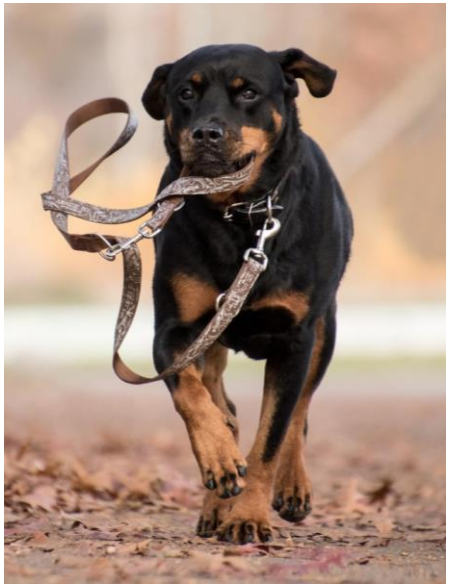
Data sampling

- Sample data into 3 groups
  - Train – Adjust trainable model parameters to minimize a loss function on the training data.
  - Validate – Evaluate the loss function on the test data and adjust hyperparameters as needed.
  - Test – Final evaluation of performance on the test data for reporting.
- This strategy ensures a robust model and accurate performance estimation.



Types of training

- Supervised learning
  - All data is labelled.
  - Regression and classification.
- Unsupervised learning (clustering)
  - No data is labelled.
  - Classification, but classes have no names.
- Semi-supervised learning.
  - Small amount of labelled data.
  - Synthesize more labelled data to train with.



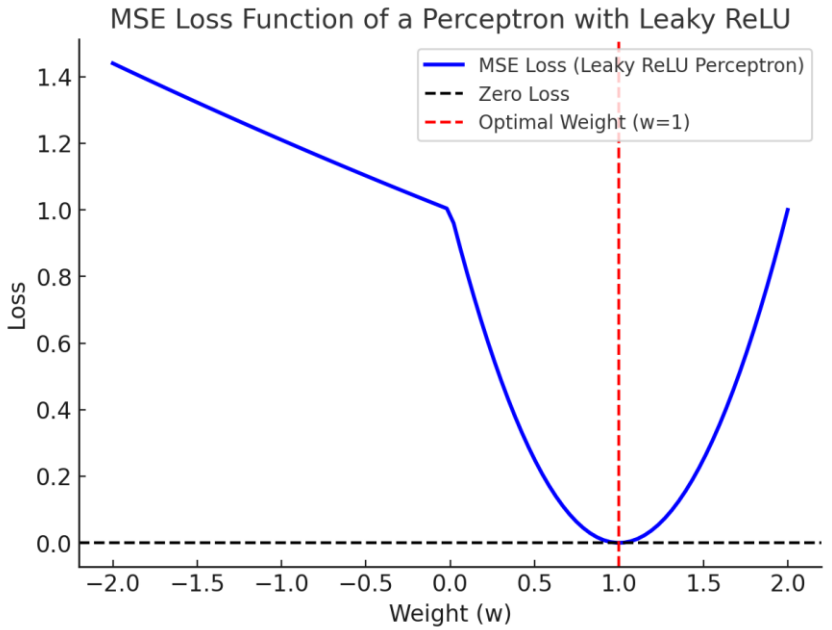
Semi-supervised learning for dogs

What happens to the model during training?

1. Define a “loss function” – how far predictions are from labels.
2. Parameterize the loss function on the trainable parameters.
3. Find the parameters which minimize the loss (optimize).
    - Analytical solutions for simple problems like linear regression.
    - Iterative gradient descent for complex problems.

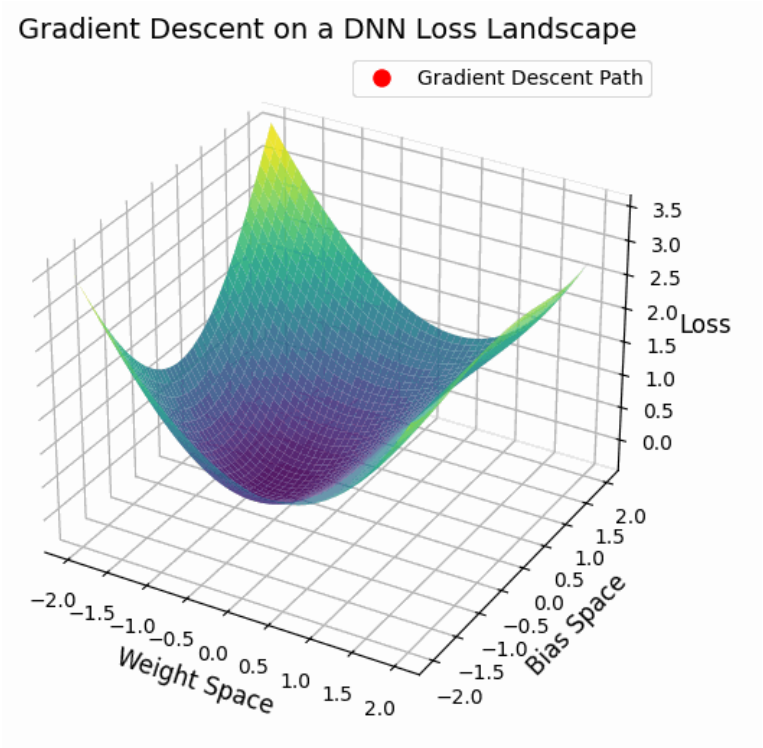
The loss surface

- A hyper-dimensional function mapping parameters to a “loss” value.
  - Regression: Mean Squared Error.
  - Classification: Binary Cross Entropy.
- Millions of parameters in the domain.
- Goal: Find the parameters with the minimum loss value.



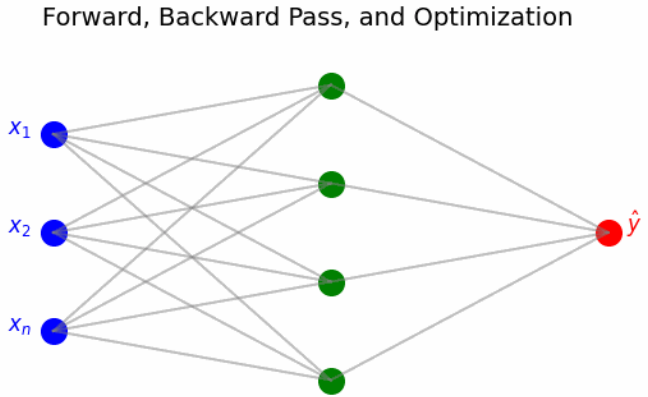
Optimizing on the loss surface

- Gradient descent techniques attempt to minimize the loss, given labelled data.
- Full evaluation of the loss function and its gradient is intractable.
- Optimization can have pitfalls.
  - Local minima.
  - Flat or irregular areas.



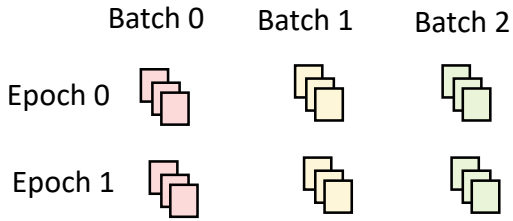
Forward-backward propagation

- An iterative way to compute gradient descent.
- Forward pass makes a prediction and calculates the loss.
- Backwards pass computes the gradient using chain rule.
- Average the gradient over several samples.
- Update the parameters using the average gradient and an optimization strategy.

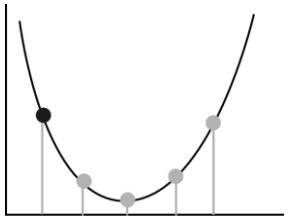


Main knobs to control training process

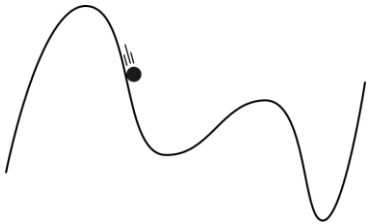
- Epochs – Number of training runs through the data.
- Batch size – How many data samples per parameter update.



- Learning rate – How much to change the parameters each update. Usually, the learning rate is adaptive.



- Momentum – Allows the optimizer to escape shallow minima.



Other strategies to control training

- Regularization – Penalizes extreme parameter values which can fit to noise.
- Dropout rate – Deactivate some neurons from parameter updates.
- Early termination – Track if learning rate is plateauing.
- Batch normalization – Normalize activation inputs to stabilize training.

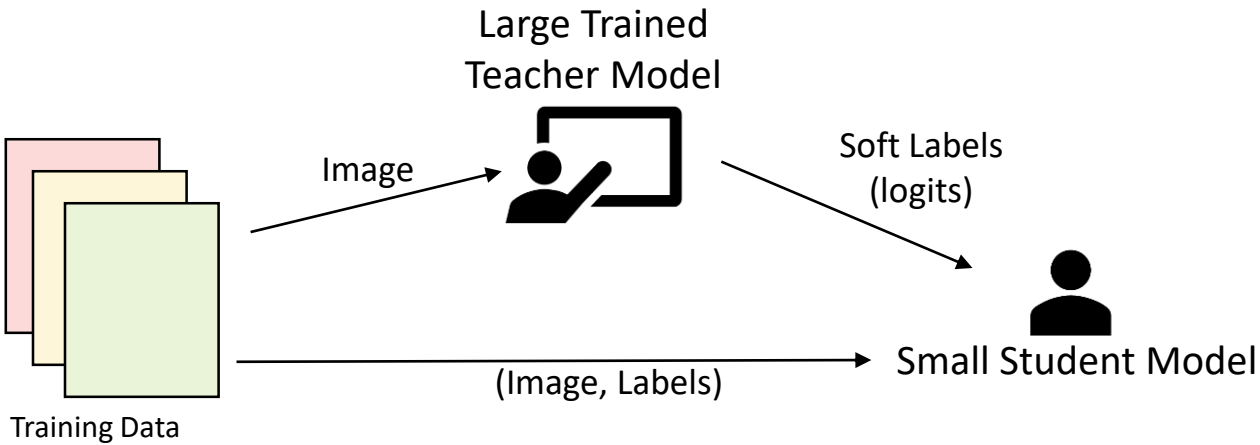
Adapting model to a task

- Use existing model (foundational) as a starting point to reduce training workload.
- Transfer learning – A new output layer (head) is created and trained. **Freeze** weights of the backbone.

- Fine tuning – Retraining the model with some layer weights **unfrozen** so they can be trained.

Adapting model to a task

- Knowledge Distillation – Allows a small model to learn complexities of the data from a larger teacher. An important tool for embedded vision!



Metrics to monitor during training

General Machine Learning Metrics:

- Loss curve – Tracks the calculated loss as parameters get updated.
- True positive (TP) – Detected a class and it was present.
- True negative (TN) – Did not detect a class and it was not present.
- False positive (FP) – Detected a class and it was not present.
- False negative (FN) – Did not detect a class and it was present.
- Accuracy – Percentage of correct predictions.
- Precision – Percentage of positive detections that were correct.
- Recall – Percentage of actual positives that were detected.

Metrics to monitor during training

General Machine Learning Metrics:

- F1 Score: $2 \cdot \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$
- Confusion Matrix – Tracks correct vs. predicted labels.

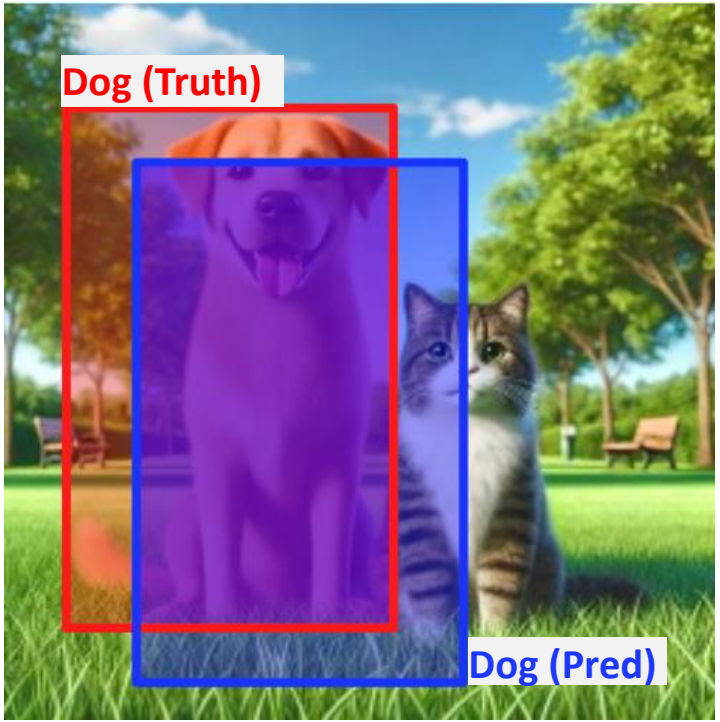
Actual \ Predicted	A	B	C
A	41	3	5
B	4	12	9
C	2	1	5

Metrics to monitor during training

Object detection metrics:

- Intersection-over-union (IoU) – What percentage of the bounding box is correct?

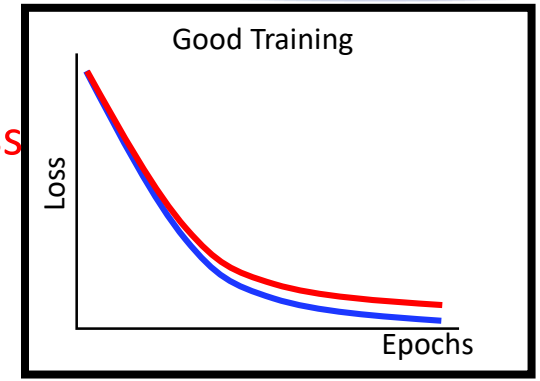
$$\text{IoU} = \frac{\text{Intersection}}{\text{Prediction} + \text{Ground Truth} - \text{Intersection}}$$

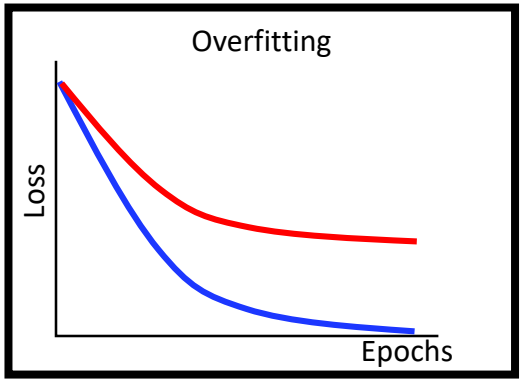


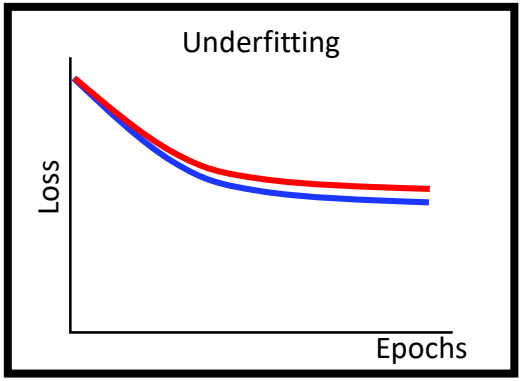
When to stop training

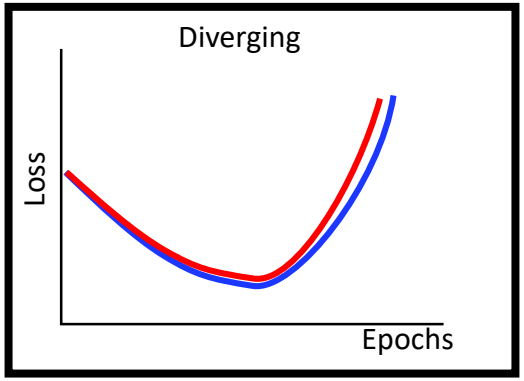
Training Loss

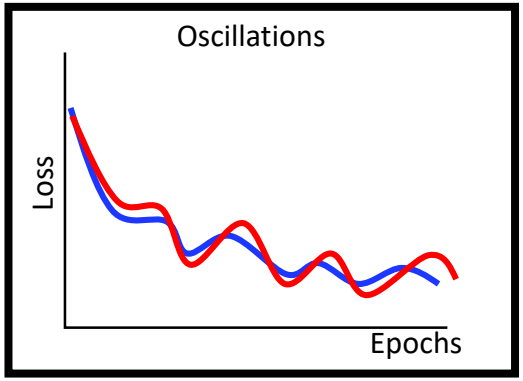
Validation Loss

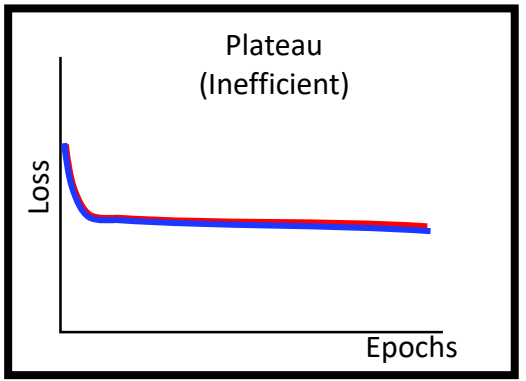












Solutions to problems that arise when training

Overfitting – The model learns non-generalizable aspects of the training data.

- Reduce model complexity – Less capability to learn from noise.
- Data augmentation – Ensure training data is more general.
- Regularization – Less capability to learn specific patterns.
- Dropout – Zeros out random parameters to encourage learning across the network.

Solutions to problems that arise when training

Underfitting – The model cannot understand the data sufficiently.

- Increase model complexity – Increased ability to model features.
- Train longer – Seeking lower loss function values.
- Improve or add features – This adds additional information it can use.
- Improve data quality – Noisy or irrelevant data can confuse the model training.

Solutions to problems that arise when training

Low Precision or Recall – indicates imbalanced priorities of the model.

- Re-balance training set – Add many examples of a particular scenario to reinforce.
- Weighted loss functions – Provide higher effect of scenarios we want to reinforce.
- Improve or add features – Custom tailored features that highlight certain properties of the data, for example color space conversions.

Solutions to problems that arise when training

Inefficient learning – Convergence is slow or unstable.

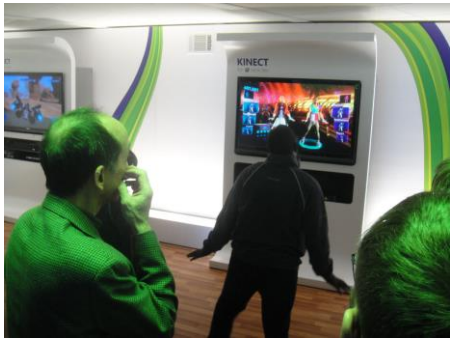
- Adaptive learning rate – adjust the learning rate to converge quickly, but avoid oscillations.
- Batch normalization – Ensure that effect size of each data is applied consistently in the learning; allows higher learning rate.
- Transfer learning – Shortcut the initial learning process by using weights from another model.
- Increased batch size – Training with more data at a time could allow higher learning rates.

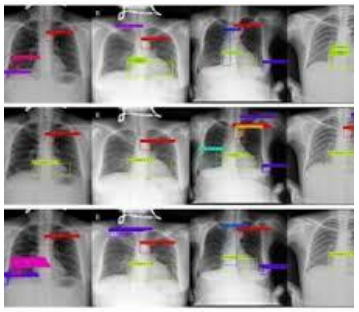
Common training frameworks and datasets

- TensorFlow – Powerful, widely supported [https://www.tensorflow.org/]
- TensorFlow Lite (now LiteRT) – Embedded inference [https://ai.google.dev/edge/litert]
- PyTorch – Easy to get started and widely known [https://pytorch.org/]
- ONNX / ONNX Runtime – Portable model format [https://onnxruntime.ai/]
- TensorRT – Optimization framework for NVIDIA hardware [https://developer.nvidia.com/tensorrt]
- Kaggle - AI and ML public data sets [https://www.kaggle.com/]
- HuggingFace – Models and datasets [https://huggingface.co/]

Conclusions

- Deep neural networks combined with computer vision are one of the most powerful tools for sensing.
- Due to their “black box” nature, we must use metrics to guide the learning process towards an acceptable result.
- This is a field with many ideas and practitioners! Many problems you will encounter already have answers.





Resources

3Blue1Brown : Deep learning video series

https://www.3blue1brown.com/topics/neural-networks

Practical Deep Learning for Coders

https://course.fast.ai/

PyTorch Lightning

https://lightning.ai/