Dent.py: Training Dental Models from Zero to Hero

Part 3 of 4

Training the Model

What we done so far?

Explored the dataset and identified needed transformations

Defined a problem, determined the task

Chose our inputs and outputs

Built a pipeline covering:
- Model
- Optimizer
- Loss Functions
- Evaluation Metrics

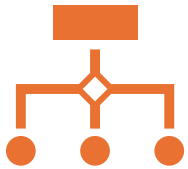# Objectives

- Identify Essential Resources
  - Gain insight into the tools, technologies, and strategies required to successfully train models.

- Navigate Challenges in Model Training
  - Explore techniques to address common obstacles
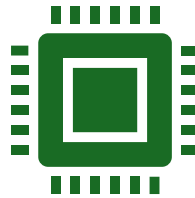 - Optimize performance through experimentation and tracking.

How to train your model?

What Affects Training?

Parameters

Control the model's learning process, such as hyperparameters and architecture choices.

Resources

Computational power, including GPUs, CPUs, memory, and storage, determines the training speed and scalability.

Time

Training duration influences performance, convergence, and experimentation cycles.

Hyperparameters

Hyperparameters

- Configurations set before training that control how the model learns
- Tuning
  - Process of selecting the best set of hyperparameters for a model to optimize its performance on a given task.
- Examples
  - Learning Rate
  - Batch Size
  - Optimizer

Hyperparameters - Strategies

Grid Search

Test a predefined range of values for each hyperparameter.
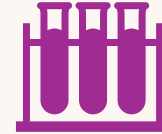
Random Search

Sample hyperparameters randomly within a range for faster exploration.

Manual Tuning

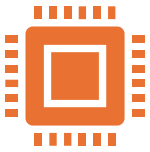Adjust based on insights, experiment results, or prior knowledge.

Tools to Use

Weights & Biases (Wandb): Track experiments and visualize results.

Resources

Resources

Computational Power:

GPUs: Specialized for parallel processing, crucial for deep learning

CPUs: Versatile, but slower for training large-scale models.

Memory:

High-capacity RAM and VRAM for handling large datasets and model architectures.

Storage:

Fast access to datasets and models, requiring SSDs.

Energy:

Training can be resource-intensive, requiring efficient power setups.

Why GPUs?

- Parallel Processing

- Faster Training Times

- Scaling Up

GPU vs CPU

Aspect	CPU	GPU
Architecture	Few cores optimized for sequential tasks.	Thousands of cores for parallel tasks.
Performance	Better for single-threaded tasks (data preprocessing, I/O).	Superior for multi-threaded tasks (training neural networks).
Flexibility	General-purpose computation.	Specialized for compute-intensive tasks.
Cost	Low	High

CPU vs GPU

| Aspect	CPU	GPU
Architecture	Few cores optimized for sequential tasks.	Thousands of cores for parallel tasks.
Performance	Better for single-threaded tasks (data preprocessing, I/O).	Superior for multi-threaded tasks (training neural networks).
Flexibility	General-purpose computation.	Specialized for compute-intensive tasks.
Cost	Low	High

What to Look for in a GPU?

Memory (VRAM)
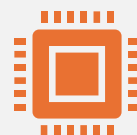
Tensor Cores

Bandwidth

Price

GPU Services

Cloud Providers

GCP

AWS

Azure

SaaS

Google Colab

Kaggle

DIY

Build your own machine

Cloud Services

- Renting GPUs through cloud service providers like AWS, Google Cloud, or Azure.
- Advantages:
  - Flexible: Pay-as-you-go pricing.
  - Scalable: Access to high-performance GPUs for large models.
  - No Hardware Maintenance: Avoids setup and repair costs.
- Disadvantages:
  - Very Costly for long-term use.
- Examples:
  - Google Cloud: NVIDIA A100, V100 instances.
  - AWS: EC2 P3, P4 instances.
 - Azure: NCv4 Series for AI workloads.

SaaS Platforms

- Free tools that provide limited access to GPUs for training small to medium models.

- Google Colab:
  - Free tier offers access to GPUs like Tesla T4 and K80.
  - Pro/Pro+ tiers increase resources with faster GPUs and longer runtimes.

- Kaggle:
  - Integrated with NVIDIA P100 GPUs for free training.
  - Well-suited for experimentation and competition setups.

- Advantages:
  - Free or low-cost.
  - Beginner-friendly environment with pre-installed frameworks.

- Disadvantages:
  - Limited GPU time and memory.
  - Not ideal for large-scale projects.
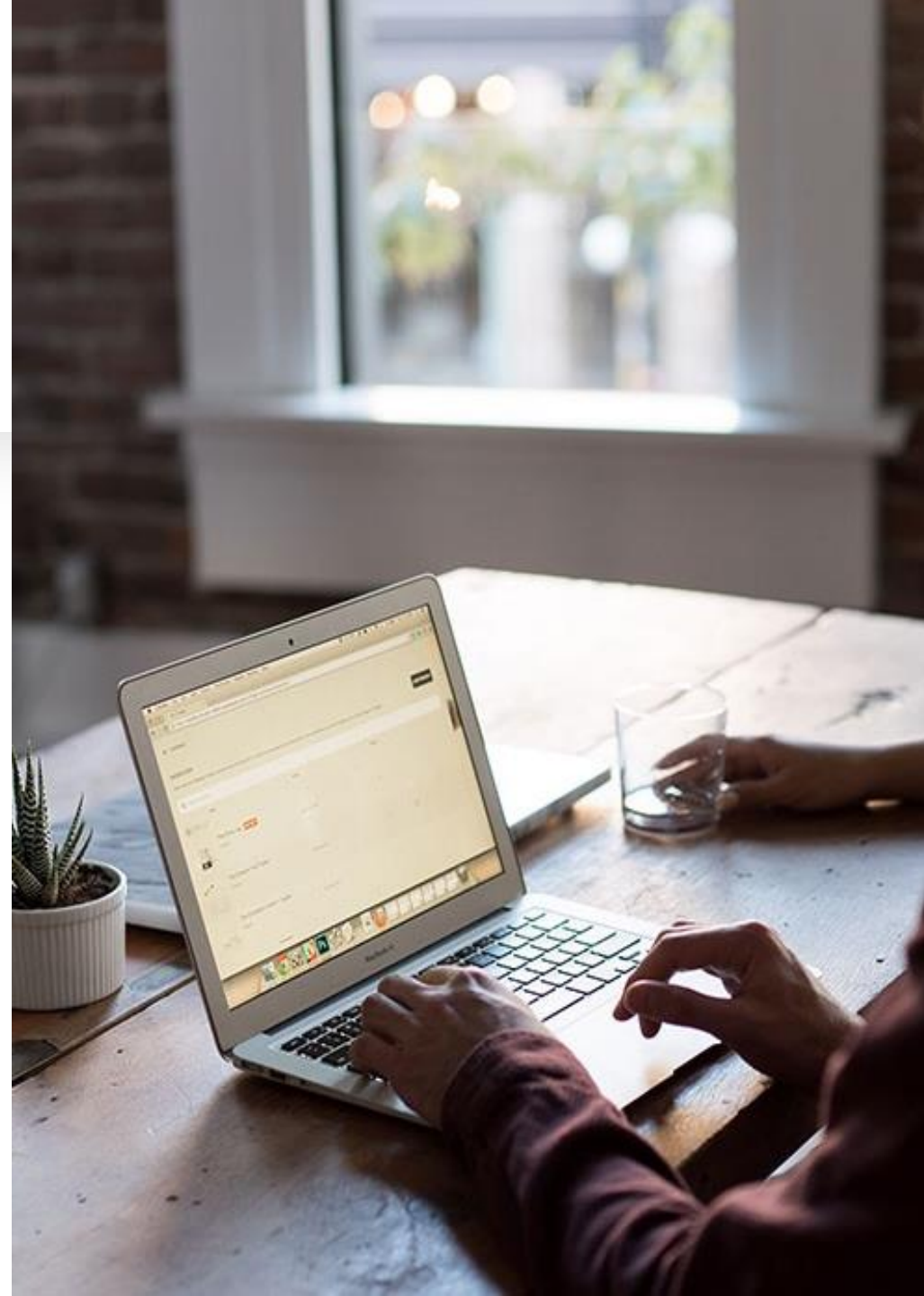
DIY Setup

- Building or buying a PC with a dedicated GPU for training.
- Advantages:
  - Full control over hardware.
  - No recurring cloud costs.
  - Offline training capability.
- Disadvantages:
  - High initial cost.
  - Maintenance and upgrades are your responsibility.
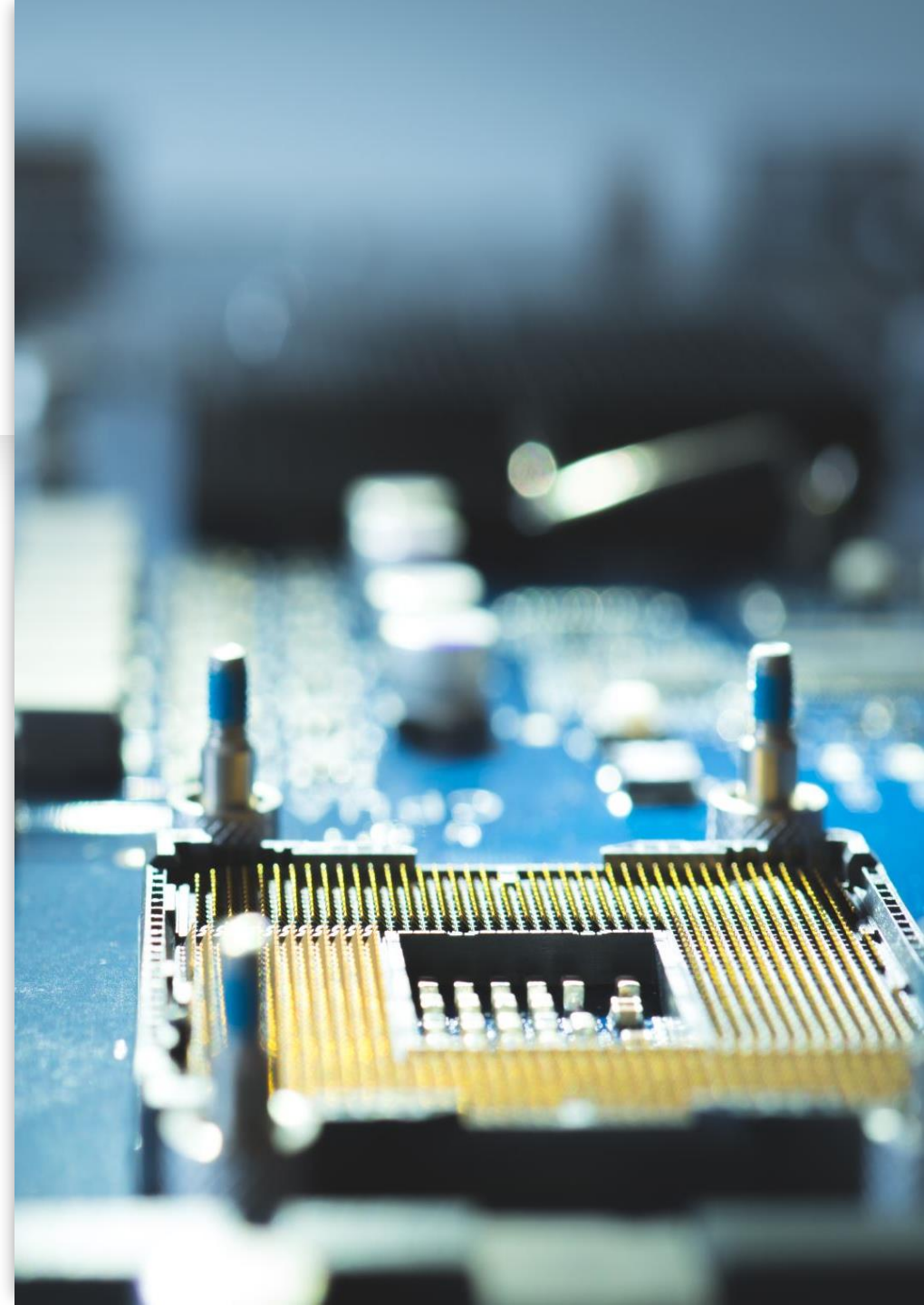
Feeling a Little Overwhelmed?

- At Knights-AI, we've got you covered!
    - Successfully built **two high-performance machines** for AI training.
    - Learned the ins and outs of DIY setups, from choosing GPUs to optimizing performance.
- Want to Build Your Own?
    - Check out our comprehensive guide on building a custom training machine.
    - Step-by-step instructions, tips, and recommendations for every budget.

Time

Why Time Matters

Convergence

Training time impacts how well the model learns patterns from the data.

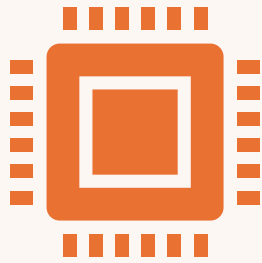Longer training can lead to better performance but risks overfitting.

Experimentation

Time determines how many experiments you can run to test ideas.

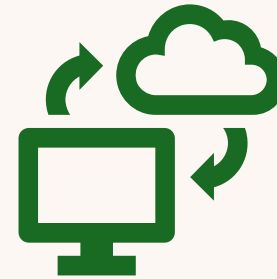Faster iterations accelerate research and development cycles.

Why Time Matters

Hardware Dependency

Training time depends heavily on the computational resources (GPUs, CPUs).

Optimizing code and hardware utilization can significantly reduce time.
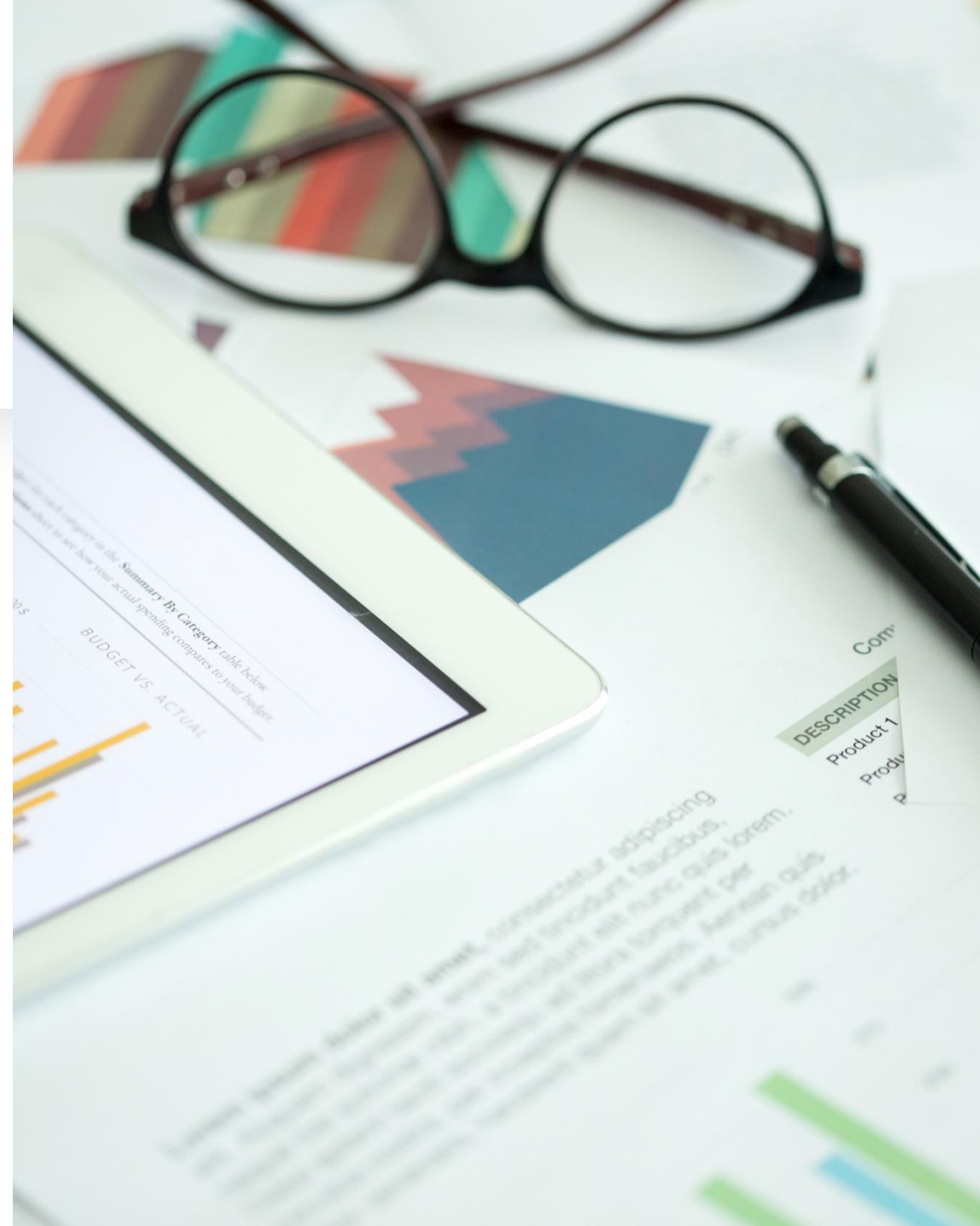
Energy and Cost:

Training large models for extended periods can be expensive in energy and cloud costs.

Checkpointing

- Checkpointing saves the current state of a model during training

- It includes:
  - Weights
  - Biases
  - Optimizer states.

Why It's Crucial

Recovery from Interruptions

Resume training from the last saved checkpoint in case of hardware failures, power outages, or crashes.

Time-Saving

Avoid starting training from scratch if an experiment is interrupted.

Fine-Tuning and Experimentation

Use saved checkpoints to adjust hyperparameters or fine-tune without restarting the process.

Tracking Progress

Save checkpoints periodically to monitor improvements and choose the best-performing model.

Checkpointing – Best Practices

01
Save checkpoints **regularly** (e.g., every epoch or every few hours).

02
Include **metadata** (training loss, validation accuracy, epochs) for easier tracking.

03
Store checkpoints **efficiently**, using formats like **.pth** or **.h5** to minimize file size.

Final Note

- At this stage, you've built a solid foundation in:
  - **Data Preprocessing**: Cleaning and preparing data for training.
  - **Data Augmentation**: Enhancing datasets for better generalization.
  - **Problem and Task Definition**: Clearly understanding your objective.
  - **Input and Output Selection**: Identifying the data format for your model.
  - **Pipeline Building**: Designing workflows for efficient training.
  - **Model Training**: Applying the techniques to achieve optimal performance.
- You're now equipped with the knowledge to tackle real-world challenges!

Questions?

Workshop Activity

- Notebooks Link
  - https://github.com/KnightsLab/EMRA-Workshop