Applied Workshop for “Mathematical Foundation of Machine Learning”

Ido Ben-Shaul and Yuval Zelig

June 2023

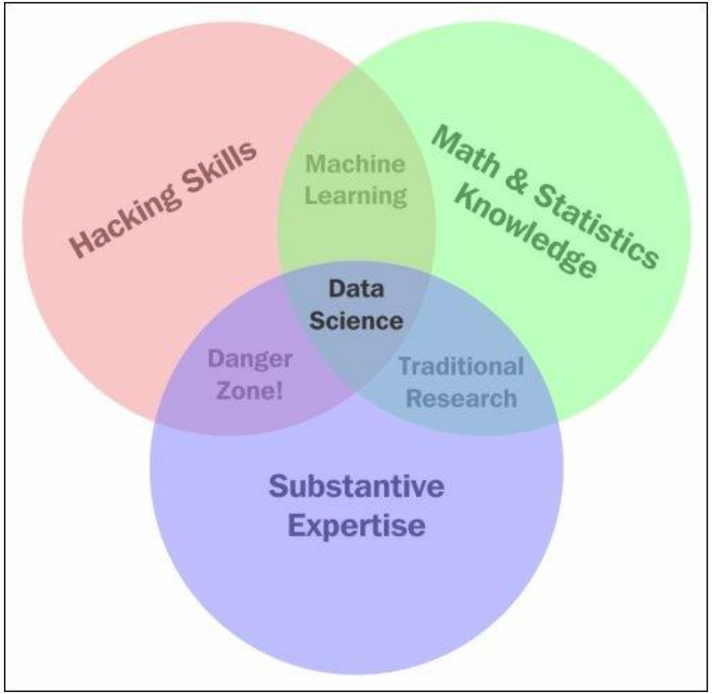
TEL AVIV UNIVERSITY

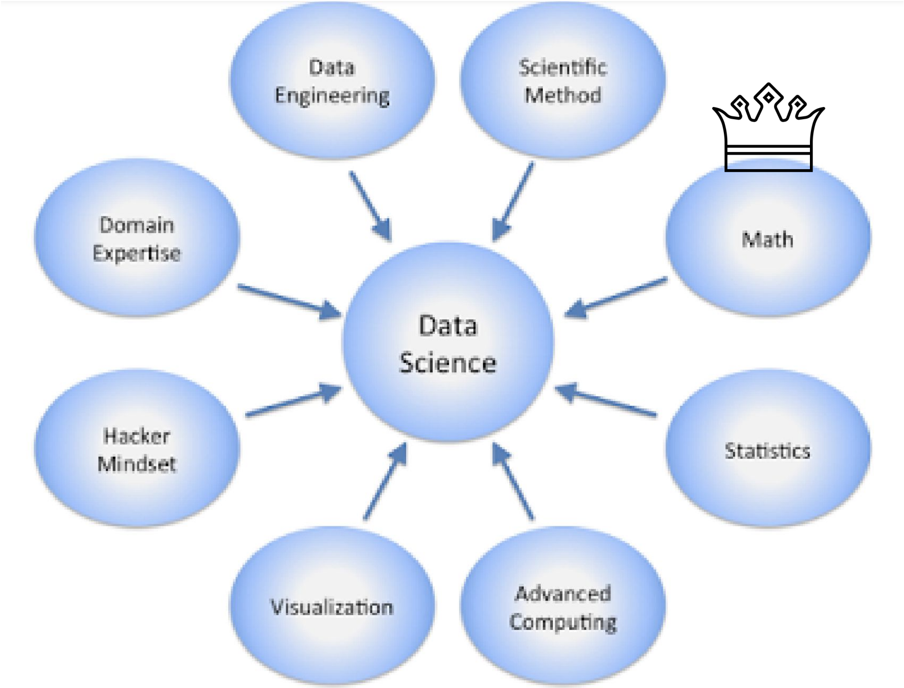
Basic Outline

- Introduction
- 1st Section - Compute Facilities:
  - AWS - Linux Image
  - Jupyter Notebook/Lab
  - Python and package Management: pip, Conda
  - Git – downloading open source projects
- 2nd Section - Classic Machine Learning and Tools
  - NumPy, Pandas, SciPy, etc..
  - Sklearn – Classifiers, Regressors, Feature Importance
  - Wavelet Forest + xgboost
- 3rd Section – Sparsity Probe, Neural Collapse and Beyond
  - Deep Learning Frameworks:
    - Pytorch + torchvision
    - Pytorch Lightning
    - MLOps - Weights and Biases
    - Transformers - HuggingFace
  - Our code
    - SparsityProbe - using Index vs. Norms(with constant \alpha)
    - Geometric Wavelet Decomposition
    - NeuralCollapse + SVSL

Introduction

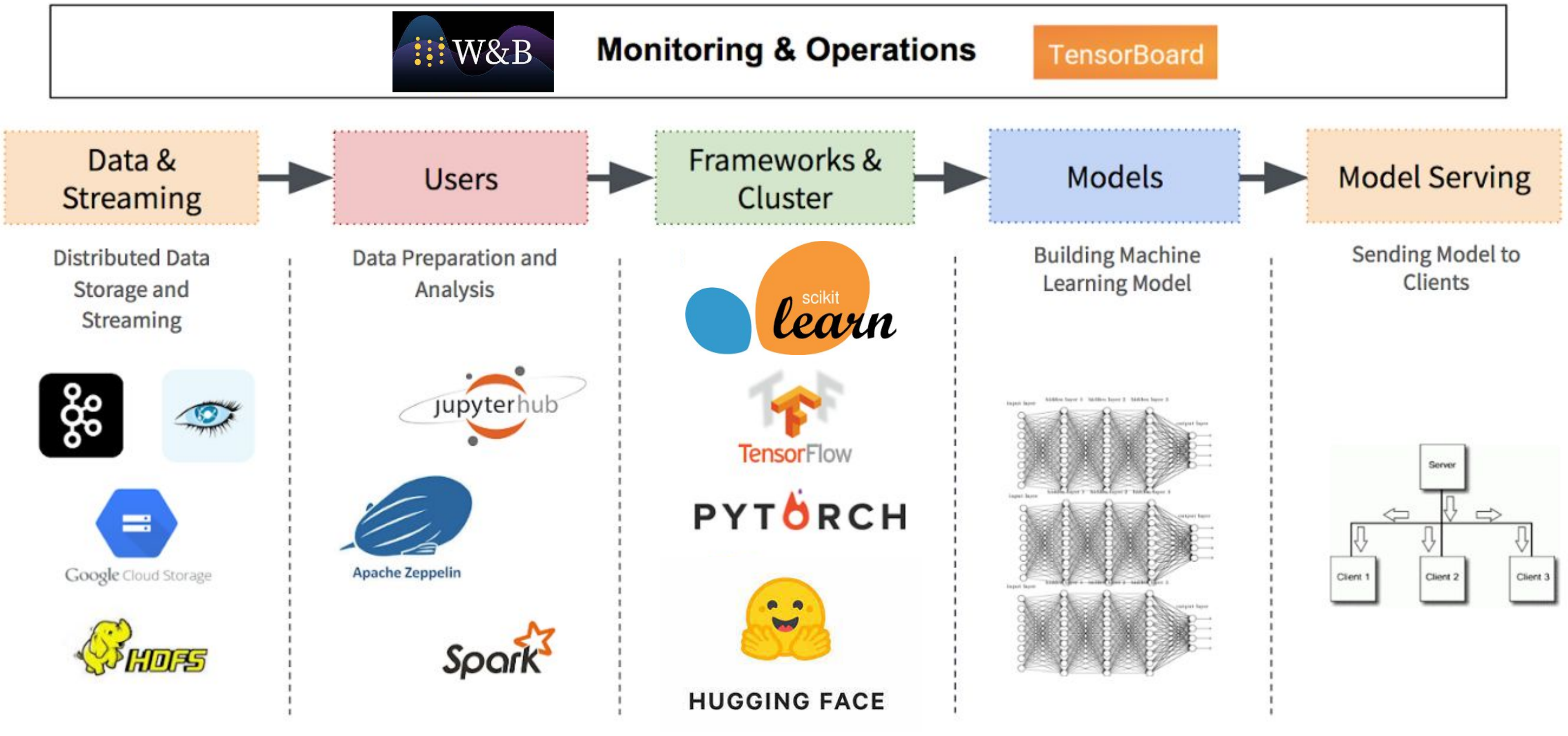
Machine Learning





https://www.altoros.com/blog/the-challenges-of-operating-a-machine-learning-model/
https://www.knowledgehut.com/blog/data-science/data-science-venn-diagram

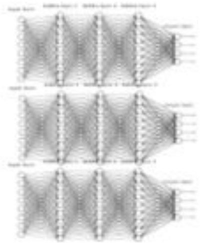
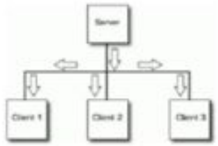
Machine Learning Pipelines



https://www.altoros.com/blog/the-challenges-of-operating-a-machine-learning-model/

Data Science Pipeline on DC/OS

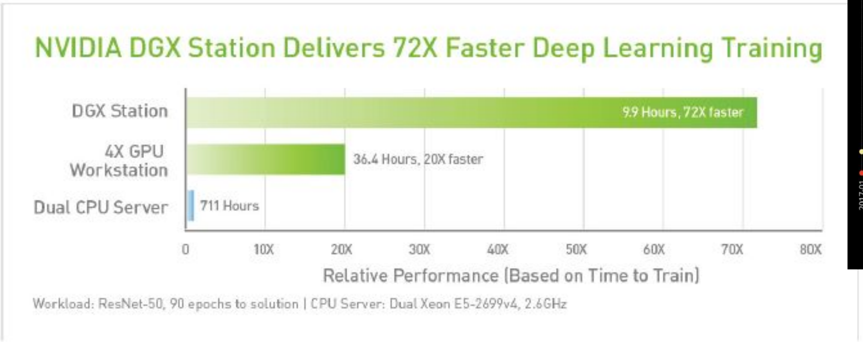
Monitoring & Operations — DATADOG — Tensorboard

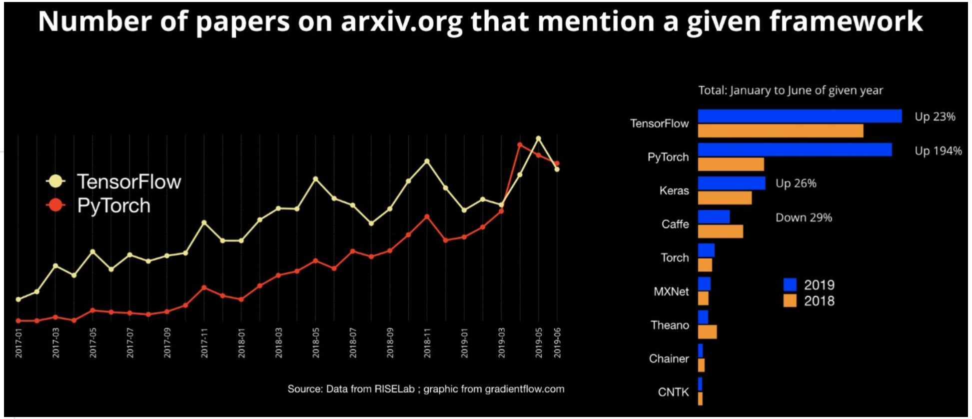
Data & Streaming	Model Engineering	Model Training	Model Management	Model Serving
Distributed Data Storage and Streaming	Data Preparation and Analysis	Distributed Training using Machine Learning Frameworks	Storage of trained Models and Metadata	Use trained Model for Inference
HDFS	jupyterhub Apache Zeppelin Spark	TensorFlow Apache Spark PYTORCH mxnet		

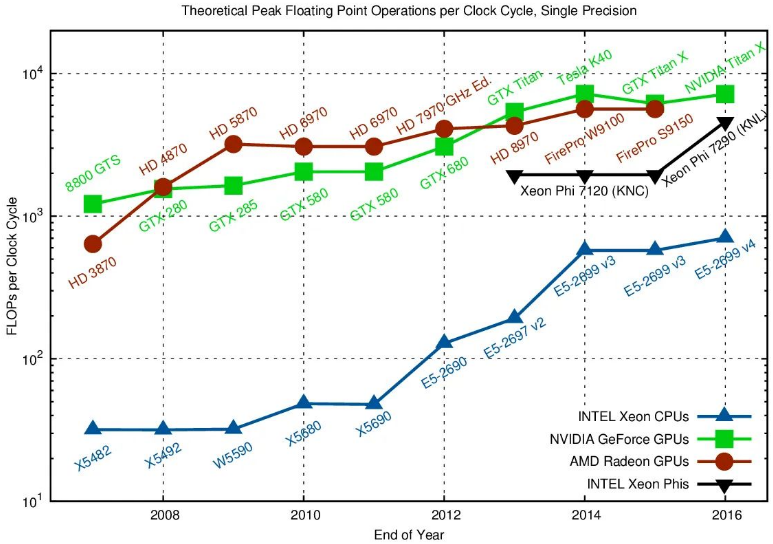
Continuous Integration — Jenkins

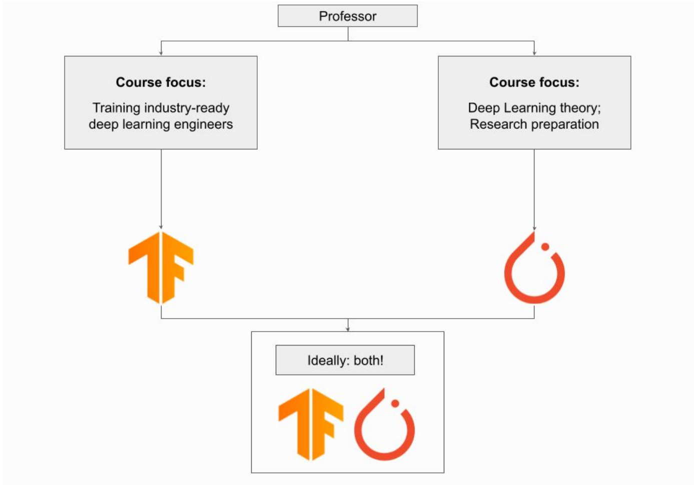
DC/OS Management — kubernetes — MESOS

Why GPU? Why PyTorch?







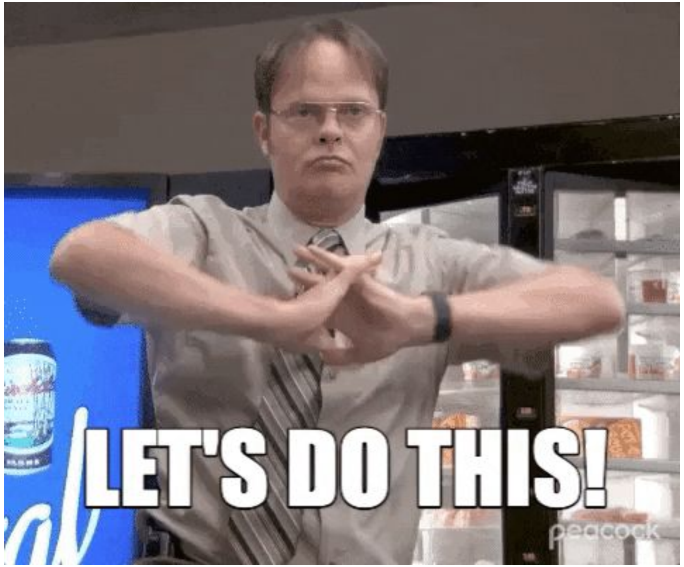


https://developer.nvidia.com/deep-learning-frameworks

https://arxiv.org/pdf/1810.12210.pdf

Course Projects

- Ability to use course knowledge and build something hands-on
    - Get into ML/DL
    - See Data-Science from a mathematical perspective
    - Try to understand the inner workings of modern models!
- Compute resources
- Support from Yuval, Ido, Shai



Compute Resources

Setting up AWS image

- Instances Panel
- AMIs -> Launch instance from AMI -> Course Image
    - For high compute resources (with GPU) g4dn.2xlarge is great/ smaller machines can also work
    - Instance Details: https://aws.amazon.com/ec2/instance-types/ (GPU is under 'Accelerated Computing')
    - Starting/Stopping Machine - **Remember to stop machine when not using it!**

Using a terminal – recommended software

- Windows
  - MobaXTerm - https://mobaxterm.mobatek.net/
- Mac
  - iterm - https://iterm2.com/
  - Regular terminal
- Linux
  - terminal should do the job ☺️

Setting up AWS image

- Setting up ssh - how to connect: https://www.youtube.com/watch?v=50PMYG_l0Us
    - Download key-pair
    - ssh-keygen -t dsa #(will make folder ~/.ssh)
    - mv ~/Downloads/<key-pair> to ~/.ssh
    - cd ~/.ssh
    - chmod 400 ~/.ssh/<key-pair>
    - ssh -i ~/.ssh/<key-pair> <username>@<Public IPv4 address>
    - ssh -i ~/.ssh/<key-pair> <username>@<Public IPv4 address> -L <local_port>:localhost:8888
        - E.g. ssh -i ~/.ssh/ido_key_pair.pem ubuntu@54.198.203.51 -L 8892:localhost:8888
- Conda Enviorment – all of your python packages are in your conda env
    - Download key-pair
    - Activate conda enviorment - source activate <conda_env_name>
        - We use: pytorch_p38
    - Check conda environments - conda env list
    - Check installed packages – conda list
    - Check installed packages for pip – pip list

https://docs.conda.io/projects/conda/en/latest/
https://pip.pypa.io/en/stable/

Sanity Checks

- python – make sure we’re on python 3.8
  - import torch
  - import SparsityProbe
  - torch.cuda.is_available() - for GPU users
- Check the following folders exist
  - /home/ubuntu/projects/MFOML_CourseExamples
  - /home/ubuntu/projects/SparsityProbe

Using Jupyter-Lab

- Activating Jupyter Lab
  - `source activate <conda_env_name>`
  - Sanity Check –
    - `python`
    - `import torch; import SparistyProbe`
    - `exit()`
  - `cd ~/.ssh`
  - `jupyter-lab`
  - Copy http://localhost:<your_forwarding_port>/lab?token=<token> and put in browser – you should see your aws machine
- Jupyter-Lab is very simple to use
  - Using terminal - remember to activate your environment if you want your thing to run 😊
  - New notebook should already be using your enviorment. Each notebook you can:
    - Run python code
    - Run bash commands

Classic ML

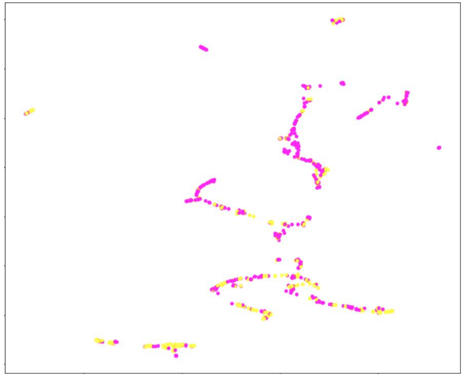
In Notebook – workshop_examples.ipynb

- Basic Jupyter Functionalities
  - Loading data example – torchvision
  - Plotting with plotly
  - Autograd - https://github.com/omniscientoctopus/Physics-Informed-Neural-Networks

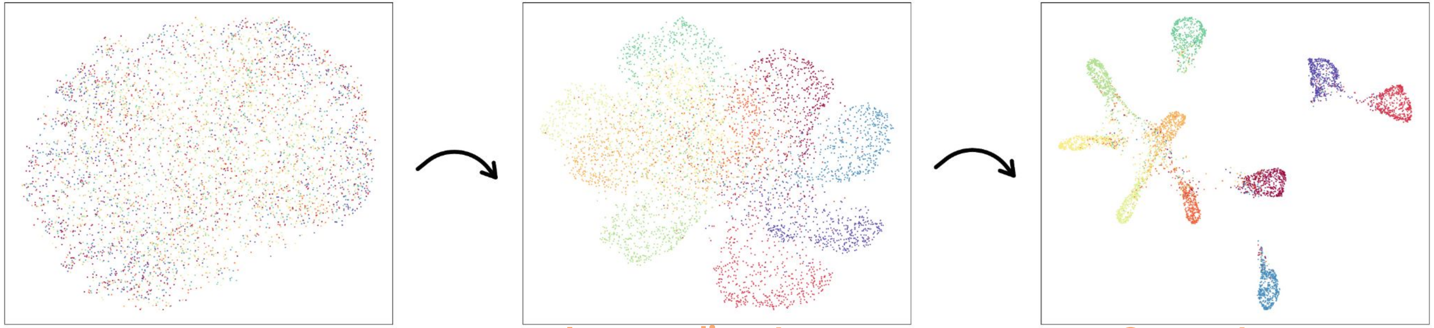
- Example – Wine Dataset
  - Loading data with pandas
  - Training basic ML models on data
    - Wavelet Forest
    - DecisionTreeRegressor
    - XGBRegressor
    - RandomForestRegressor

ML vs. DL

- Good separability in input feature space → ML
- All successful Machine Learning algorithms look for this geometry:
  - Support Vector Machines, Random Forest, Gradient Boosting, etc.
- **If not, can we transform to a better feature space through feature engineering/deep learning (CNN, Resnets, Transformers etc)?**



MNIST DATASET



Input Layer | Intermediate Layer | Output Layer

[1] UMAP of trained ConvNet on MNIST Dataset - Ben-Shaul, I. and Dekel, S., “Sparsity-Probe: Analysis tool for Deep Learning Models”, <i>arXiv e-prints</i>, 2021.
[2] F. Chollet, Deep Learning with Python, Manning, Nov. 2017.

Decision Trees

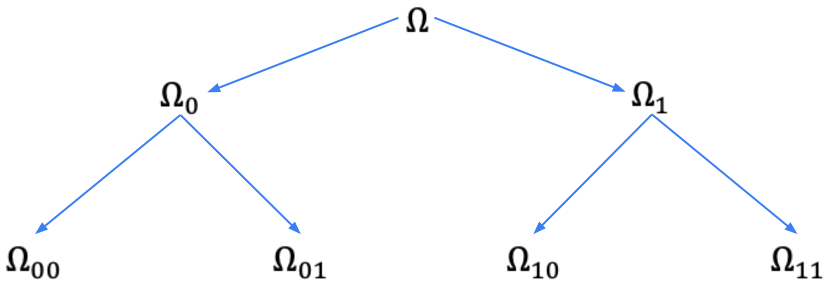
In the functional setting we are given a function

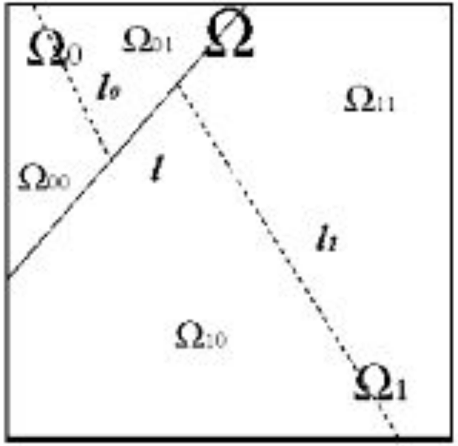
$$f \in L_2(\Omega), \Omega \subset \mathbb{R}^n$$

In applications, point values (or even "density")

$$x_i \in \Omega, \qquad f(x_i)\,, i \in I$$

We apply recursive subdivision of the data





Decision Trees

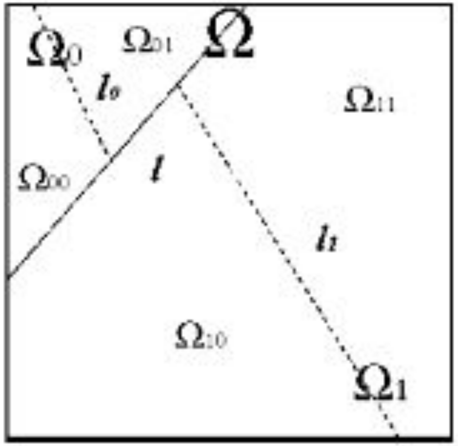
Minimizing Variance: ‘Optimal’ subdivisions in lower dimensions: using minimizing hyper-surface cuts and least-squares polynomials $Q_{\Omega'}, Q_{\Omega''} \in \Pi_{r-1}(\mathbb{R}^n)$

$$\sum_{x_i \in \Omega'} \left(f(x_i) - Q_{\Omega'}(x_i)\right)^2 + \sum_{x_i \in \Omega''} \left(f(x_i) - Q_{\Omega''}(x_i)\right)^2 \qquad \Omega' \cup \Omega'' = \Omega$$

Piecewise constants

$$Q_{\Omega'}(x) = C_{\Omega'} := \frac{1}{\#\{x_i \in \Omega'\}} \sum_{x_i \in \Omega'} f(x_i)$$

$$Q_{\Omega''}(x) = C_{\Omega''} := \frac{1}{\#\{x_i \in \Omega''\}} \sum_{x_i \in \Omega''} f(x_i)$$



Decision Tree Inference

For new input $x = (x_1, \dots, x_n)$

$$\tilde{f}(x) \coloneqq Q_{\Omega'}(x)$$

where,

I. $x \in \Omega'$

II. $\Omega' \in T$, is a leaf

Random Forest

- 'Best' decision tree: NP-hard problem!
- Goal: overcome the 'greedy nature' of a single tree.
- 'Bagging': For each j, we select a random subset X^j consisting of 80% of the input data points.
- Over each random subset we create a tree T_j

- Each tree provides

$$\tilde{f}_j(x) := Q_{\Omega'}(x), x \in \Omega \text{ is a leaf}, \Omega \in T_j$$

- Random Forest: $\tilde{f}(x) := \frac{1}{J}\sum_{j=1}^{J} \tilde{f}_j(x)$

- Adding trees → Convergence to minimal risk [Breiman 2001]

Geometric Wavelet

Let Ω' be a child of Ω in tree T

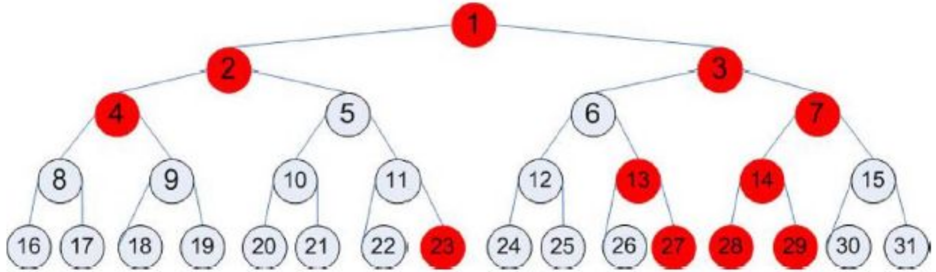
$$\psi_{\Omega'}(x) \coloneqq \mathbb{1}_{\Omega'}(x)\big(Q_{\Omega'}(x) - Q_{\Omega}(x)\big) \qquad \|\psi_{\Omega'}(x)\|_p = \|Q_{\Omega'}(x) - Q_{\Omega}(x)\|_{L_p(\Omega')}$$

Under simple conditions: $f = \sum_{\Omega \in T} \psi_{\Omega}$

We get a ranking: $\|\psi_{\Omega_1}\|_2 \geq \|\psi_{\Omega_2}\|_2 \geq \|\psi_{\Omega_3}\|_2 \geq \cdots$

M-term geometric wavelet sum:

$$S_M(f) \coloneqq \sum_{m=1}^{M} \psi_{\Omega_M}$$



Variable/Feature Importance

The wavelet-based VI is derived by imposing a restriction on the adaptive re-ordering of the wavelet components (11), such that they must appear in 'feature related blocks'. To make this precise, let $\{x \in \mathbb{R}^n, f(x)\}$ be a dataset and let $\tilde{f}$ represent the RF decomposition, as in (8). We evaluate the importance of the i-th feature by

$$S_i^\tau := \frac{1}{J} \sum_{j=1}^{J} \sum_{\Omega \in \mathcal{T}_j \cap V_i} \|\psi_\Omega\|_2^\tau, \quad i = 1, \ldots, n, \tag{20}$$

where, $\tau > 0$ and V_i is the set of child domains formed by partitioning their parent domain along the ith variable. This allows us to score the variables, using the ordering $S_{i_1}^\tau \geq S_{i_2}^\tau \geq \cdots$. Recall that our wavelet-based approach transforms classification problems into the functional setting (see section 2) by mapping each label l_k to a vertex $\vec{l}_k \in \mathbb{R}^{L-1}$ of a regular simplex. Therefore, in classification problems, the wavelet norms in (20) are given by (7) which implies that we provide a unified approach to VI.

Deep Learning Frameworks

PyTorch Basics

- Very easy to use, and large community, constant updating, easily dynamic
- Torch “tensor”

```
[3]: import torch
     import torchvision
     import torch.nn as nn
     import numpy as np
```

```
[9]: a = np.random.rand(20)
     a
```

```
[9]: array([0.84674746, 0.73806255, 0.62570047, 0.16766903, 0.11829172,
            0.60357139, 0.01697733, 0.47920269, 0.38706433, 0.0294812 ,
            0.67079046, 0.45913126, 0.54982041, 0.05621114, 0.85943097,
            0.71611574, 0.67799682, 0.05094388, 0.5749863 , 0.00195263])
```

```
[10]: torch.tensor(a)
```

```
[10]: tensor([0.8467, 0.7381, 0.6257, 0.1677, 0.1183, 0.6036, 0.0170, 0.4792, 0.3871,
              0.0295, 0.6708, 0.4591, 0.5498, 0.0562, 0.8594, 0.7161, 0.6780, 0.0509,
              0.5750, 0.0020], dtype=torch.float64)
```

Creating a network

[29]:
```python
class ConvLayer(nn.Module):
    def __init__(self, input_size: int, output_size: int):
        super().__init__()
        self.input_size = input_size
        self.output_size = output_size
        self.conv = nn.Conv2d(self.input_size, self.output_size, kernel_size=3, stride=1, padding=1)
        self.bn = nn.BatchNorm2d(self.output_size)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.conv(x)
        x = self.bn(x)
        x = self.relu(x)
        return x
```

[11]:
```python
class ConvArch(nn.Module):
    def __init__(self, input_channel_number: int, input_image_width:int, num_classes: int, depth: int, width: int):
        super().__init__()
        self.input_channel_number = input_channel_number
        self.input_image_width = input_image_width
        self.num_classes = num_classes
        self.depth = depth
        self.width = width
        self.init_layers()

    def get_initial_layers(self):
        layers = [nn.Conv2d(self.input_channel_number, self.width, kernel_size=2, stride=2, padding=0)]
        layers.append(nn.Conv2d(self.width, self.width, kernel_size=2, stride=2, padding=0))
        layers.append(nn.BatchNorm2d(self.width))
        layers.append(nn.ReLU())
        return layers

    def init_layers(self):
        self.initial_layers = nn.ModuleList(self.get_initial_layers())
        layers = []
        for _ in range(self.depth):
            layers.append(ConvLayer(self.width, self.width))
        self.secondary_layers = nn.ModuleList(layers)
        self.fc = nn.Linear(self.width*np.power(self.input_image_width//4, 2), self.num_classes)

    def forward(self, orig):
        x = orig
        for layer in self.initial_layers:
            x = layer(x)
        first_layer_output = x
        for layer in self.secondary_layers:
            x = layer(x)
        second_layer_output = x
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x
```

Pytorch Datasets:

- Classic ML datasets
- https://pytorch.org/vision/stable/datasets.html
- https://pytorch.org/tutorials/beginner/data_loading_tutorial.html
- Implement two functions: def __len__() and def __getitem__(idx)

```
[49]: from torchvision import datasets, transforms
      transform = transforms.Compose([transforms.ToTensor(),
                                      transforms.Normalize(0.1307, 0.3081)])

      mnist_dataset = datasets.MNIST("/Users/ibenshaul/datasets/MNIST", download=True, train=True, transform=transform)
```

```
[52]: first_instance = mnist_dataset[0]
      first_instance[0].shape, first_instance[1]
```

```
[52]: (torch.Size([1, 28, 28]), 5)
```

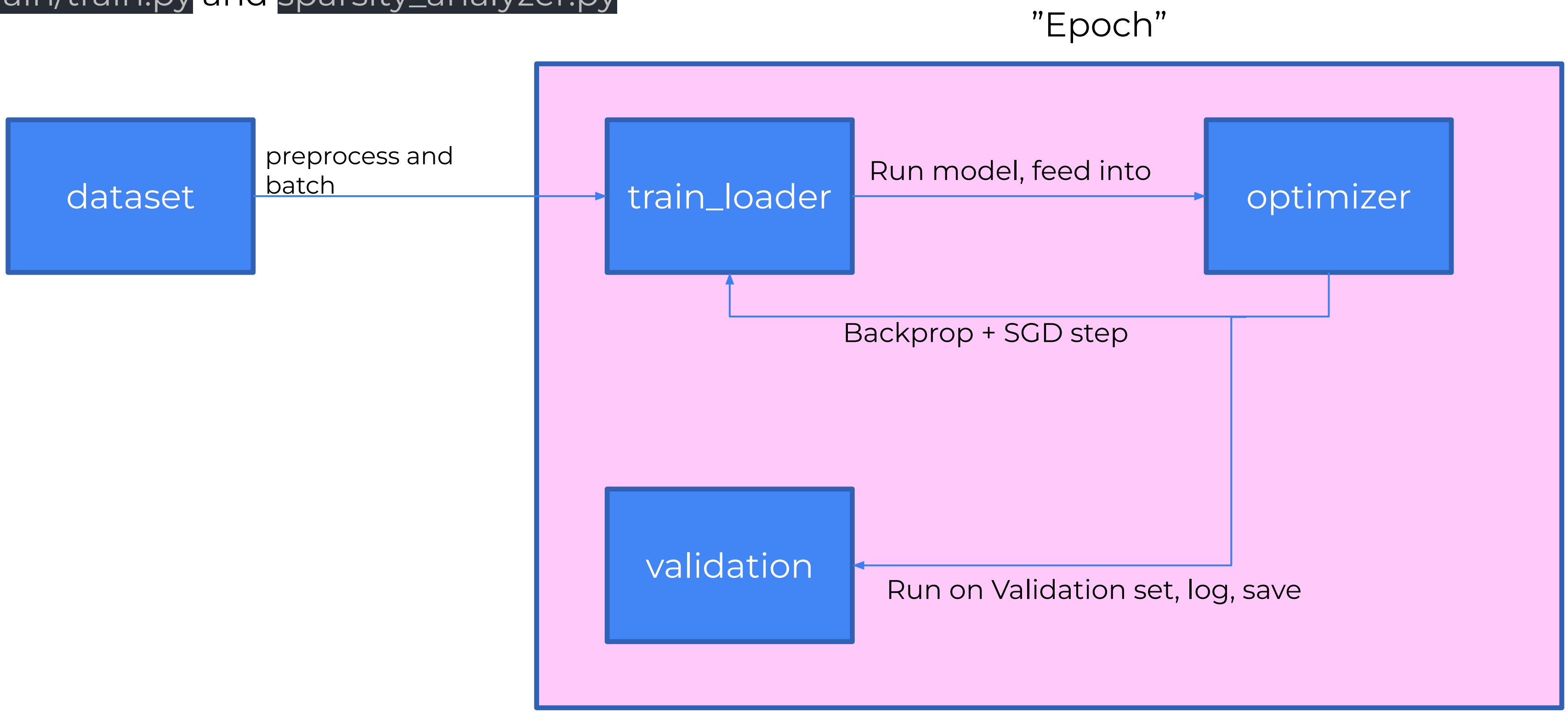
```
[53]: output = model(first_instance[0].unsqueeze(0))
      output, output.shape
```

```
[53]: (tensor([[-0.1471, -0.3543, -0.4778,  0.1071,  0.2955,  0.4135,  0.3324,  1.0057,
                -0.4160,  0.1131]], grad_fn=<AddmmBackward0>),
       torch.Size([1, 10]))
```

Basic PyTorch Building Blocks:

We'll go over the code at
/home/ubuntu/projects/MFOML_CourseExamples/VisionSparsityProbeExperiments

/train/train.py and sparsity_analyzer.py



Wavelet Sparsity

- We now define the Sparsity of a RF Wavelet Decomposition:
- Let $0 < \tau < p$ (eg. $p = 2$),

$$N_\tau(f,F) \coloneqq \frac{1}{J}\left\{\sum_{j=1}^{J}\sum_{\Omega\neq\Omega_0,\Omega\in T_j}\|\psi_\Omega\|_p^\tau\right\}^{\frac{1}{\tau}}$$

$$\lim_{\tau\to 0} N_\tau(f,T)^\tau \coloneqq \lim_{\tau\to 0}\frac{1}{J}\left\|\{\|\psi_\Omega\|_2\}_{\Omega\in T_j,1\le j\le J}\right\|_{l_\tau}^{\tau} = \{\#\Omega \in T\colon\ \|\psi_\Omega\|_2 \neq 0\}$$

Michael Elad. 2010. <i>Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing</i> (1st. ed.). Springer Publishing Company, Incorporated.

Tree Besov Smoothness

For $\alpha > 0, \;\; r > \alpha,$ $\quad \frac{1}{\tau} := \alpha + \frac{1}{p}$

$$|f|_{B_\tau^{\alpha,r}(T)} := \left\{ \sum_{\Omega \in T} \left(|\Omega|^{-\alpha} \omega_r(f,\Omega)_\tau \right)^\tau \right\}^{\frac{1}{\tau}}$$

Compare with classic Besov Spaces:

$$|f|_{B_\tau^{\alpha,r}} := \left\{ \sum_{Q \text{ Dyadic Cube}} \left(|\mathrm{Q}|^{-\alpha} \omega_r(f,\mathrm{Q})_\tau \right)^\tau \right\}^{\frac{1}{\tau}}$$

Forest Besov Smoothness

For $\alpha > 0$, $r > \alpha$, $\frac{1}{\tau} := \alpha + \frac{1}{p}$

$$|f|_{B_\tau^{\alpha,r}(F)} := \frac{1}{J}\left\{\sum_{j=1}^{J} |f|_{B_\tau^{\alpha,r}(T_j)}^{\tau}\right\}^{\frac{1}{\tau}}$$

<u>We can show:</u>

$$|f|_{B_\tau^{\alpha,r}(F)} \sim N_\tau(f,F)$$

Critical Besov Smoothness Index:

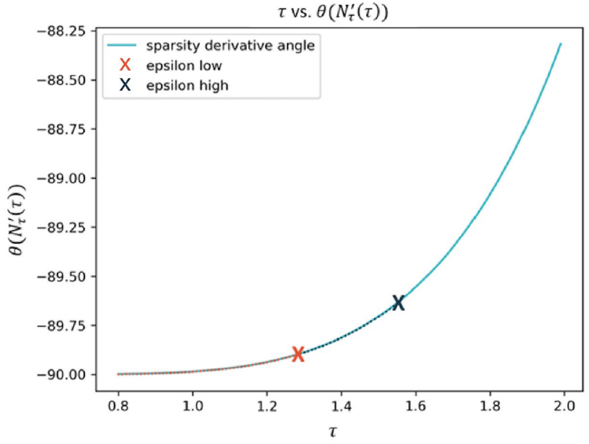
$$\tau^* := \inf_{0<\tau<2}\{\tau\colon\ N_\tau(f,F) < \infty\} \qquad \alpha^* = \frac{1}{\tau^*} - \frac{1}{p}$$

[1] Ben-Shaul, I. and Dekel, S., “Sparsity-Probe: Analysis tool for Deep Learning Models”, <i>arXiv e-prints</i>, 2021.
[2] Elisha O. and Dekel S., "Wavelet decompositions of Random Forests - smoothness analysis, sparse approximation and applications", Journal of Machine Learning Research, 2016

Approximating τ^*

1. Let f be a function, F its forest decomposition and $k \in \mathbb{N}$, $\varepsilon_{\text{high}}, \varepsilon_{\text{low}} \in \mathbb{R}$
2. Take equally spaced $\tau_k \in (0, p)$
3. Approximate

$$N'_\tau(\tau_k) := \frac{\partial N_\tau(f, F)}{\partial \tau}(\tau_k)$$



4. Take angles of derivatives:

$$\theta(\tau_k) := \arctan\left(N'_\tau(\tau_k)\right)$$

5. Define:

$$S := \left\{\tau_k : -\frac{\pi}{2} + \varepsilon_{\text{low}} \le \theta(\tau_k) \le -\frac{\pi}{2} + \varepsilon_{\text{high}}\right\}$$

$$\Rightarrow \qquad \tau^* := \frac{1}{|S|} \sum_{\tau_k \in S} \tau_k$$

Approximation high-dim smoothness from sparse samples

- Assume we have samples of a function $f_k: [0,1]^{n_k} \to \mathbb{R}^L$
- Construct a piecewise constant approximation using a Random Forest.
- Create a Wavelet decomposition of the Random Forest.
- Model the Smoothness as Sparsity of the Forest Norms:
  - Find the critical τ^* such that $N_\tau(f, \mathcal{F}) \coloneqq \frac{1}{J}\left(\sum_{j=1}^{J} \sum_{\Omega \neq \Omega_0,\, \Omega \in T_j} \|\psi_\Omega\|_p^\tau\right)^{\frac{1}{\tau}} < \infty$

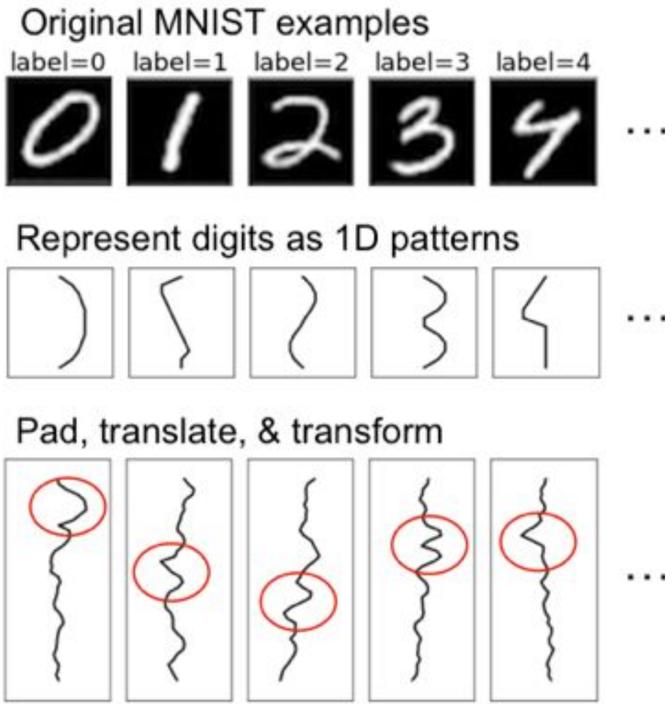
$$\tau^* \coloneqq \inf_{0<\tau<2} \{\tau \mid N_\tau(f, \mathcal{F}) < \infty\}$$

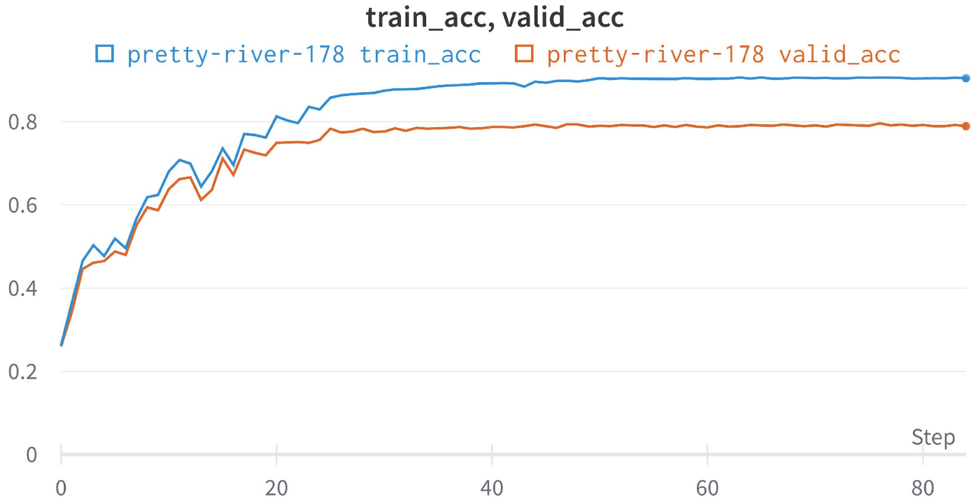
- Define the critical α-smoothness score as $\alpha^* \coloneqq \frac{1}{\tau^*} - \frac{1}{2} > 0$

[1] Ben-Shaul, I. and Dekel, S., “Sparsity-Probe: Analysis tool for Deep Learning Models”, <i>arXiv e-prints</i>, 2021.
[2] Elisha O. and Dekel S., "Wavelet decompositions of Random Forests - smoothness analysis, sparse approximation and applications", Journal of Machine Learning Research, 2016

Example – Train + SparsityProbe on MNIST1D

```
cd /home/ubuntu/projects/MFOML_CourseExamples/VisionSparsityProbeExperiments
python train/train.py --env_name 'mnist_1D_Conv_env' --epochs 85
--save_epochs --lr 0.01
```





HuggingFace

- Models
  - Example GPT2- https://huggingface.co/gpt2
- Datasets
- Spaces
- Compatibility with: PyTorch, TensorFlow, JAX, keras, etc..

HUGGING FACE

HeBERT: Pre-trained BERT for Polarity Analysis and Emotion Recognition

HeBERT is a Hebrew pre-trained language model. It is based on Google's BERT architecture and it is BERT-Base config (Devlin et al. 2018).

Hosted inference API

Text Classification | Examples

אני אוהב את ימי ראשון וסושי

Compute

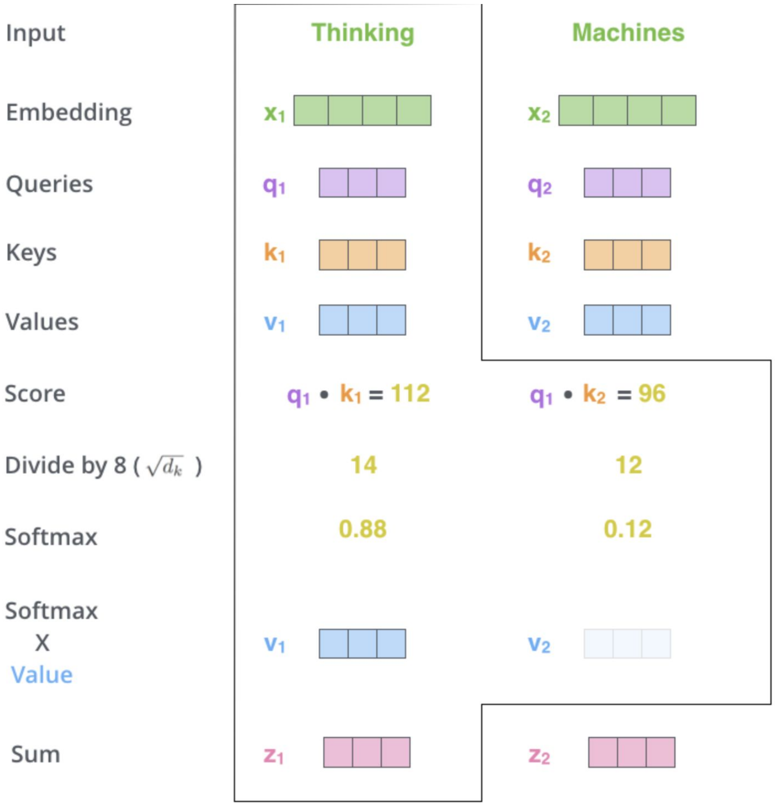
Computation time on cpu: 0.0336 s

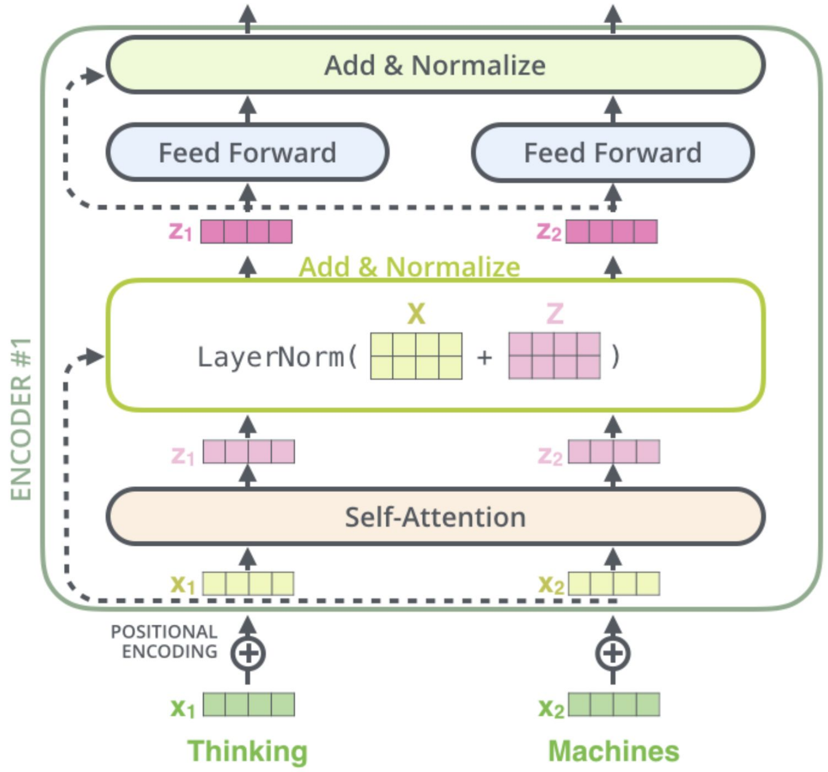
neutral	0.002
positive	0.998
negative	0.000

JSON Output | Maximize

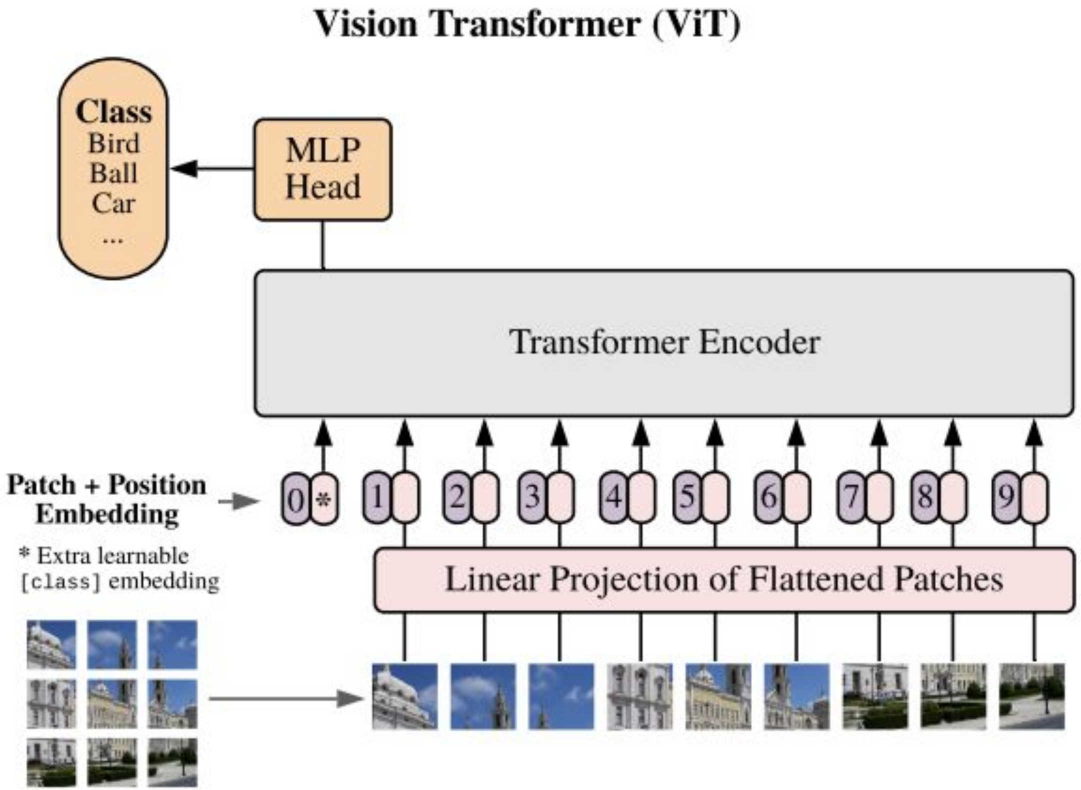
Transformers in 5 minutes

Great beginner guide - https://jalammar.github.io/illustrated-transformer/





Vision Transformers in 1 minute



[1]Dosovitskiy, A., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", <i>arXiv e-prints</i>, 2020.

Language Models – PreTraining/ Self-Supervised Learning

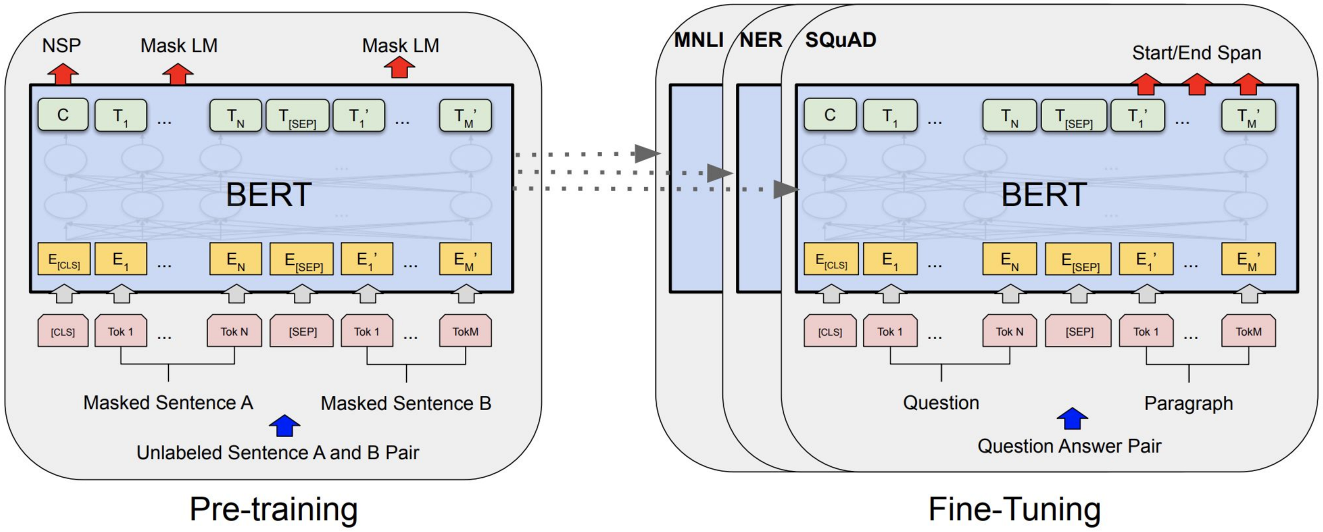
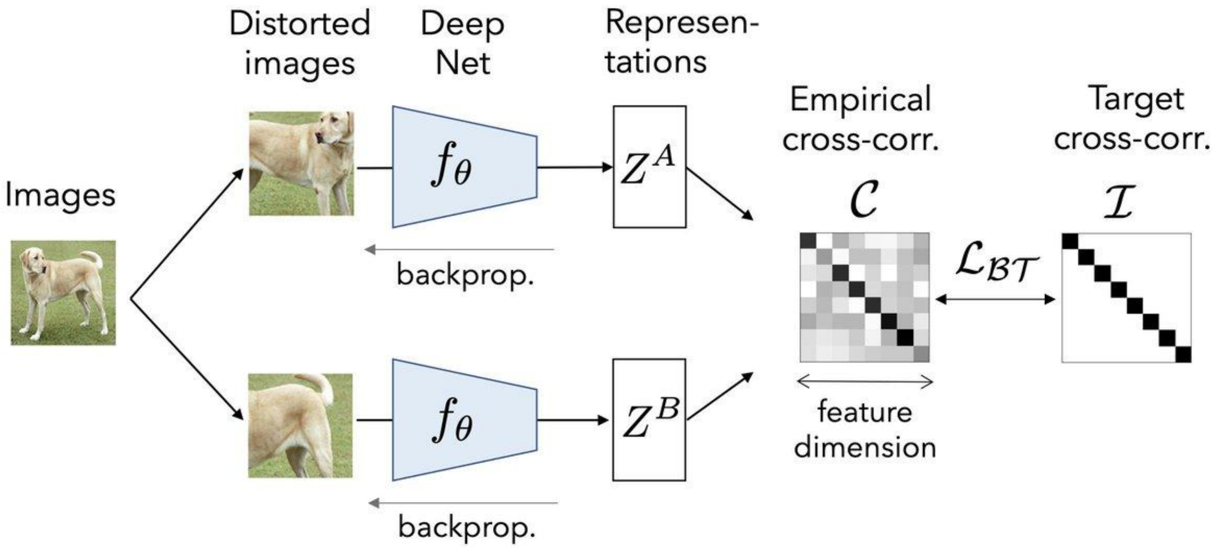


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Self-Supervised Learning in Vision



[1]Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S., “Barlow Twins: Self-Supervised Learning via Redundancy Reduction”, <i>arXiv e-prints</i>, 2021.

HuggingFace – Sequence Classification

HUGGING FACE

- https://huggingface.co/docs/transformers/training

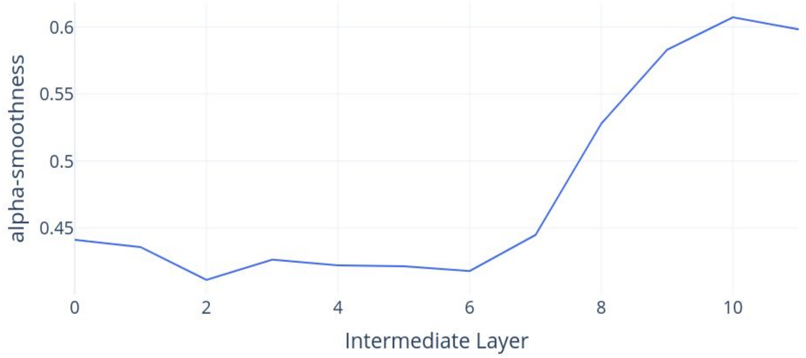
- *distilbert*: `DistilBertForSequenceClassification` (DistilBERT model)
- *albert*: `AlbertForSequenceClassification` (ALBERT model)
- *camembert*: `CamembertForSequenceClassification` (CamemBERT model)
- *xlm-roberta*: `XLMRobertaForSequenceClassification` (XLM-RoBERTa model)
- *roberta*: `RobertaForSequenceClassification` (RoBERTa model)
- *bert*: `BertForSequenceClassification` (Bert model)
- *xlnet*: `XLNetForSequenceClassification` (XLNet model)
- *flaubert*: `FlaubertForSequenceClassification` (Flaubert model)

```
model = AutoModelForSequenceClassification.from_pretrained('bert-base-uncased')
```

Example – Train Sequence-Classification with HUGGING FACE

```
cd /home/ubuntu/projects/MFOML_CourseExamples/NLPSparsityProbeExperiments
python train_glue_without_trainer.py --model_name_or_path bert-base-cased --task_name cola --num_train_epochs 3       #--use_norms
```

Corpus	|Train|	|Test|	Task	Metrics	Domain
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

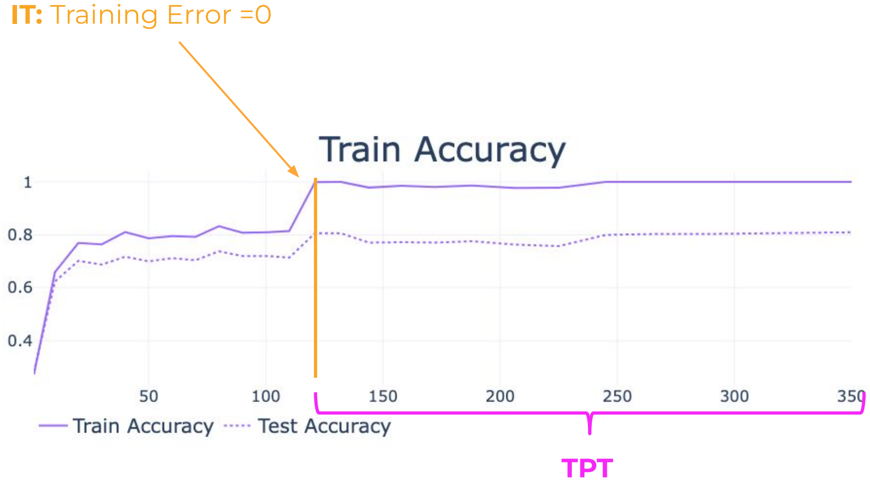


Interpolation Threshold

Example: ResNet18 trained on CIFAR10

Interpolation Threshold (**IT**): The point at which Training Error =0

Terminal Phase of Training (**TPT**)[1]: Regime after the Interpolation Threshold



[1] Papyan, V., Han, X.Y., & Donoho, D.L. (2020). Prevalence of neural collapse during the terminal phase of deep learning training. Proceedings of the National Academy of Sciences of the United States of America, 117, 24652 - 24663.
[2] Ben-Shaul, I., & Dekel, S. (2022). Nearest Class-Center Simplification through Intermediate Layers. ArXiv, abs/2201.08924.

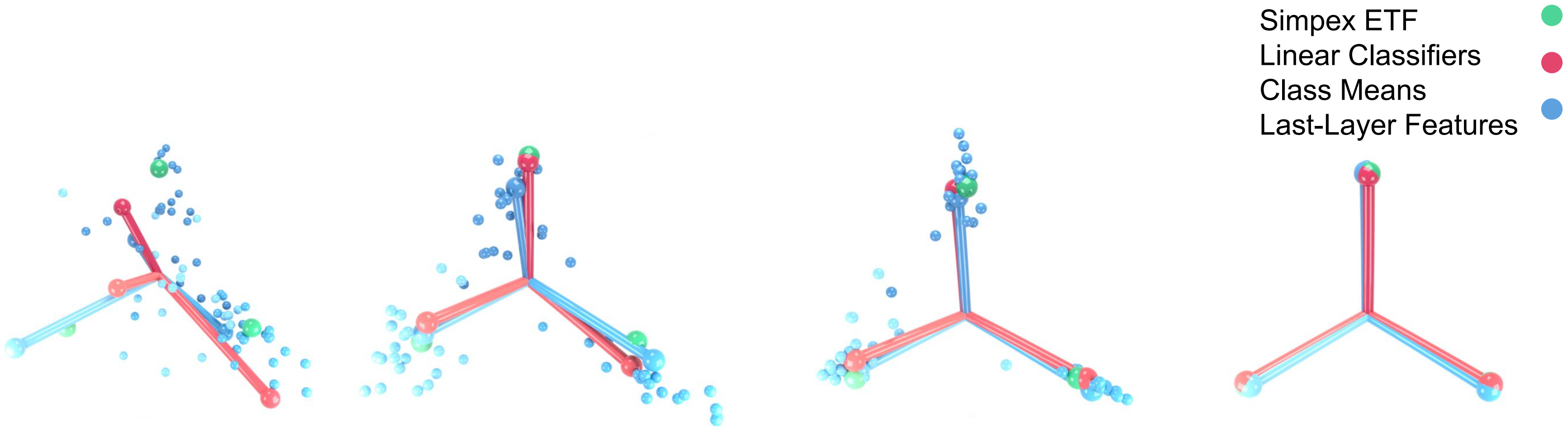
Example - Neural Collapse

(NC1) Variability Collapse: $\Sigma_W^{(k-1)} \to 0$

(NC4) Simplification to Nearest Class Center (NCC): Let

$$S \coloneqq \left\{ i \in \mathcal{I}_{\text{Train}} \,\middle|\, \arg\max_{1 \le \tilde{c} \le C} g(x_i)_{\tilde{c}} \neq \arg\min_{1 \le \tilde{c} \le C} \left\| g^{(k-1)}(x_i) - \mu_{\tilde{c}}^{(k-1)} \right\|_2 \right\}$$

then $|S| \to 0$, where $|X|$ is the number of elements in finite set X.



Papyan, V., Han, X.Y., & Donoho, D.L. (2020). Prevalence of neural collapse during the terminal phase of deep learning training. Proceedings of the National Academy of Sciences of the United States of America, 117, 24652 - 24663.

Nearest Class Center Simplification

- NCC Mismatch in intermediate layers:

$$\Lambda_{\text{Train}}^{(j)} := \frac{1}{\mathrm{N}_{\text{Train}}} \left| \left\{ \arg\max_{1 \le \tilde{c} \le C} g(x_i)_{\tilde{c}} \neq \arg\min_{1 \le \tilde{c} \le C} \left\| g^{(j)}(x_i) - \mu_{\tilde{c}}^{(j)} \right\|_2 \,\middle|\, i \in \mathcal{I}_{\text{Train}} \right\} \right|$$

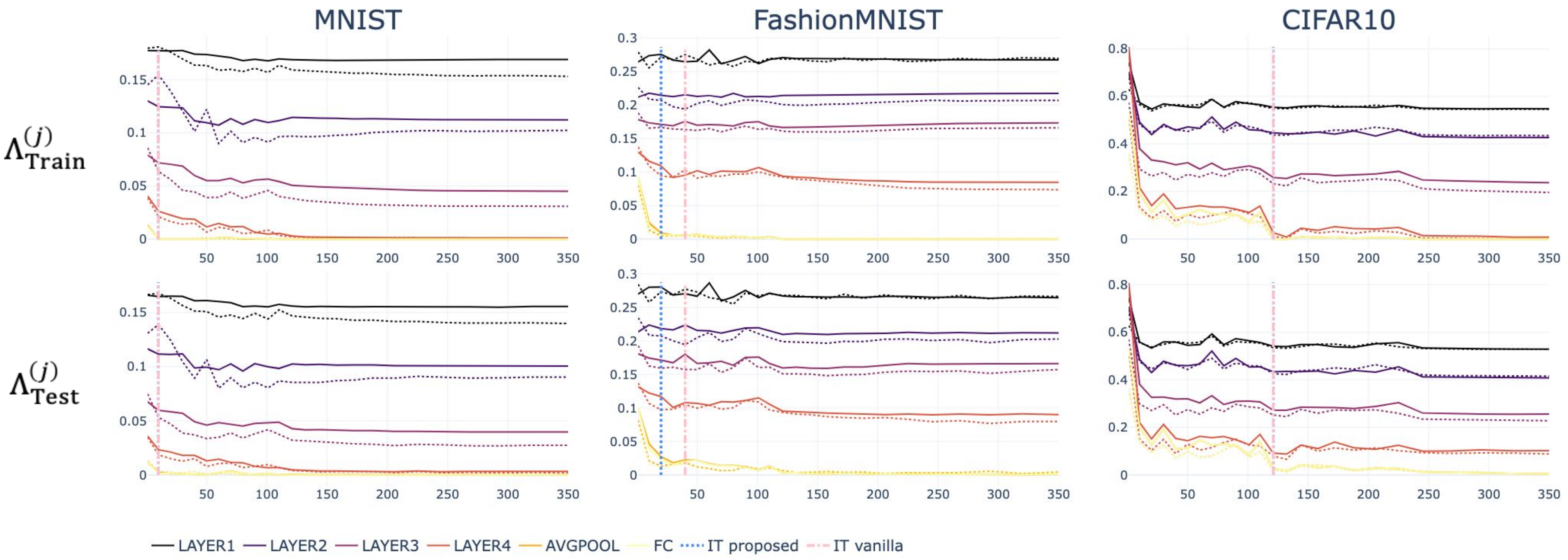
$$\Lambda_{\text{Test}}^{(j)} := \frac{1}{\mathrm{N}_{\text{Test}}} \left| \left\{ \arg\max_{1 \le \tilde{c} \le C} g(x_i)_{\tilde{c}} \neq \arg\min_{1 \le \tilde{c} \le C} \left\| g^{(j)}(x_i) - \mu_{\tilde{c}}^{(j)} \right\|_2 \,\middle|\, i \in \mathcal{I}_{\text{Test}} \right\} \right|$$

Ben-Shaul, I., & Dekel, S. (2022). Nearest Class-Center Simplification through Intermediate Layers. ArXiv, abs/2201.08924.

Nearest Class Center in Intermediate Layers

- Layer Ordering: $\Lambda^{(j)}_{\text{Train}} \geq \Lambda^{(j+1)}_{\text{Train}}$ and $\Lambda^{(j)}_{\text{Test}} \geq \Lambda^{(j+1)}_{\text{Test}}$

  NCC Improves in TPT: $\Lambda^{(j)}_{\text{Train,IT}} \geq \Lambda^{(j)}_{\text{Train,EOT}}$ and $\Lambda^{(j)}_{\text{Test,IT}} \geq \Lambda^{(j)}_{\text{Test,EOT}}$



NCC with Stochastic Variability-Simplification Loss (SVSL)

Can we encourage clustering in Intermediate Layers?

Stochastic Train class-means: for layer $l^{(j)}$, batch $\mathcal{B}$ and class c

$$\mu_{c,\mathcal{B}}^{(j)} := \underset{i \in \mathcal{B}, y_i = c}{\text{Avg}} \left\{ g_i^{(j)} \right\}$$

Stochastic Variability-Simplification Loss (SVSL): Let g be a deepnet, $\hat{y}_i = g(x_i)$ for (x_i, y_i), $i \in \mathcal{J}_{\text{Train}}$, $y_i = c$. We also let $\gamma \in \mathbb{N}, 1 \leq \gamma < k$ and $\alpha \in \mathbb{R}_+$ two hyperparameters. For batch $\mathcal{B}$ such that $(x_i, y_i) \in \mathcal{B}$, we define the SVSL:

$$L(\hat{y}_i, y_i) := \text{CE}(\hat{y}_i, y_i) + \alpha\eta \sum_{j=\gamma}^{k} \left\| g_i^{(j)}(x_i) - \mu_{c,\mathcal{B}}^{(j)} \right\|_2^2$$

NCC with Stochastic Variability-Simplification Loss (SVSL)

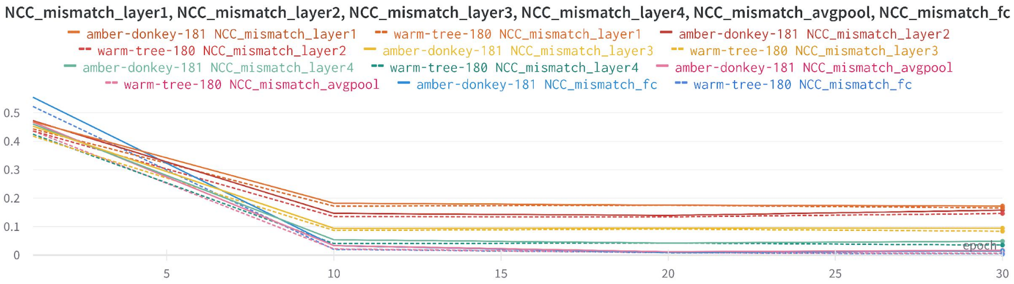
	IT		EOT		Best Test Epoch			
Dataset	Vanilla	SVSL	Vanilla	SVSL	Vanilla	In TPT	SVSL	In TPT
MNIST	**99.37**	**99.36**	99.61	**99.69**	99.65	Yes	**99.69**	Yes
Fashion MNIST	91.78	**93.13**	93.82	**93.88**	93.93	Yes	**94.03**	Yes
STL10	53.41	**55.95**	54.11	**56.65**	54.19	Yes	**56.94**	Yes
CIFAR10	**80.64**	80.56	80.96	**81.19**	80.96	Yes	**81.19**	Yes
CIFAR100	52.77	**53.28**	53.31	**54.29**	53.79	Yes	**54.29**	Yes
CoLA	51.59	**52.91**	53.46	**55.54**	53.95	No	**55.54**	Yes
RTE	**58.84**	58.12	55.23	**59.57**	**61.01**	No	60.28	Yes
MRPC	70.83	**74.26**	74.26	**75.25**	75.00	No	**76.71**	No
SST-2	87.96	**88.42**	88.42	**88.76**	**89.22**	No	**89.22**	Yes

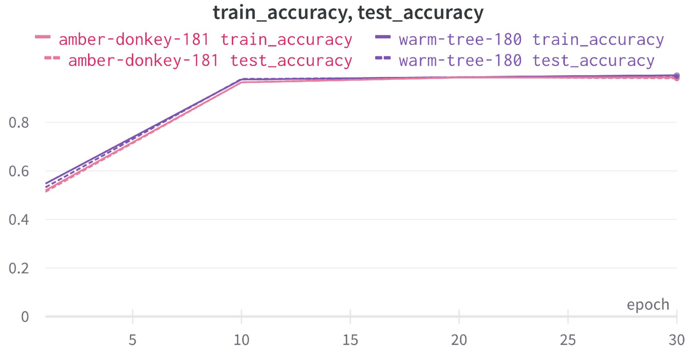
Ben-Shaul, I., & Dekel, S. (2022). Nearest Class-Center Simplification through Intermediate Layers. ArXiv, abs/2201.08924.

Neural Collapse Example – MNIST (debug mode)

```
cd NeuralCollapse/Vision/
```

- Running without SVSL (normal Cross Entropy Loss)
  - `python neuralcollapse_run.py 0 0 False configs/MNIST_Resnet18.p`
- Running with SVSL
  - `python neuralcollapse_run.py 1e-5 4 True configs/MNIST_Resnet18.p`





SP+NC Example Experiments

```
cd /home/ubuntu/projects/MFOML_CourseExamples/VisionSparsityProbeExperiments
python train/train.py --env_name 'mnist_1D_Conv_env' --epochs 85 --save_epochs --lr 0.01
```

```
cd /home/ubuntu/projects/MFOML_CourseExamples/NLPSparsityProbeExperiments
python train_glue_without_trainer.py --model_name_or_path bert-base-cased --task_name cola
--num_train_epochs 3 --use_norms
```

```
cd NeuralCollapse/
python neuralcollapse_run.py 1e-5 4 True configs/MNIST_Resnet18.p
```

Extra Materials

Useful Commands and workflows

- Nvidia-smi
- pycharm, vscode, ...
  - Developing locally and rsync
    - `rsync -r my_folder ido@54.227.191.120:/home/ubuntu/projects/`
  - Remote debugging – available on most IDEs – can be a bit complicated
- pytorch-lightning

PyTorch Lightning

https://www.pytorchlightning.ai/

PyTorch

```python
# ----------------
# TRAINING LOOP
# ----------------
num_epochs = 1
for epoch in range(num_epochs):

  # TRAINING LOOP
  for train_batch in mnist_train:
    x, y = train_batch

    logits = pytorch_model(x)
    loss = cross_entropy_loss(logits, y)
    print('train loss: ', loss.item())

    loss.backward()

    optimizer.step()
    optimizer.zero_grad()

  # VALIDATION LOOP
  with torch.no_grad():
    val_loss = []
    for val_batch in mnist_val:
      x, y = val_batch
      logits = pytorch_model(x)
      val_loss.append(cross_entropy_loss(logits, y).item())

    val_loss = torch.mean(torch.tensor(val_loss))
    print('val_loss: ', val_loss.item())
```

PyTorch Lightning

```python
# train loop + val loop + test loop
trainer = pl.Trainer()
trainer.fit(LightningMNISTClassifier())
```

Useful References

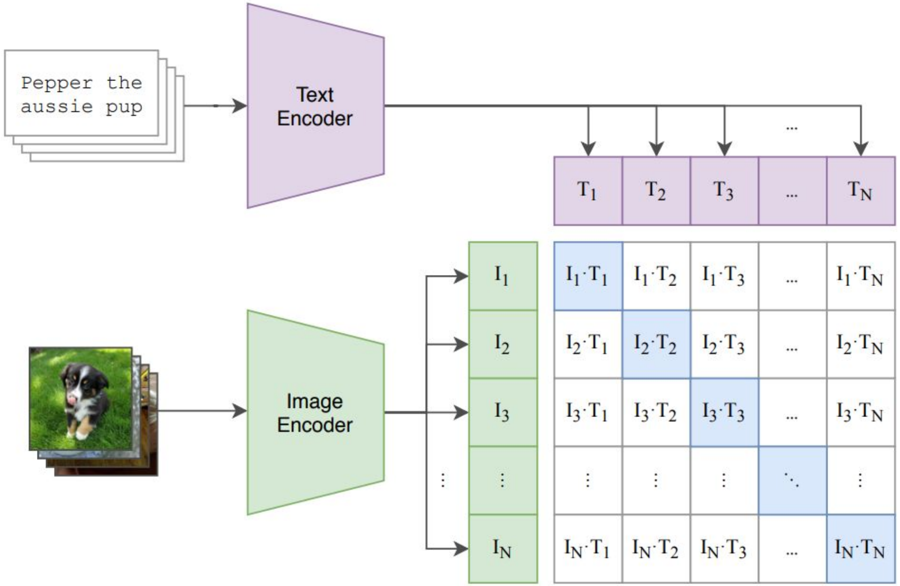
- Datasets
  - ML https://archive.ics.uci.edu/ml/datasets.html
  - DL
    - Vision: https://pytorch.org/vision/stable/datasets.html
    - NLP: https://huggingface.co/docs/datasets/index
  - Kaggle https://www.kaggle.com/datasets
- Code
  - https://github.com/idobenshaul10/MFOML_CourseExamples
  - https://github.com/idobenshaul10/SparsityProbe/tree/course_version

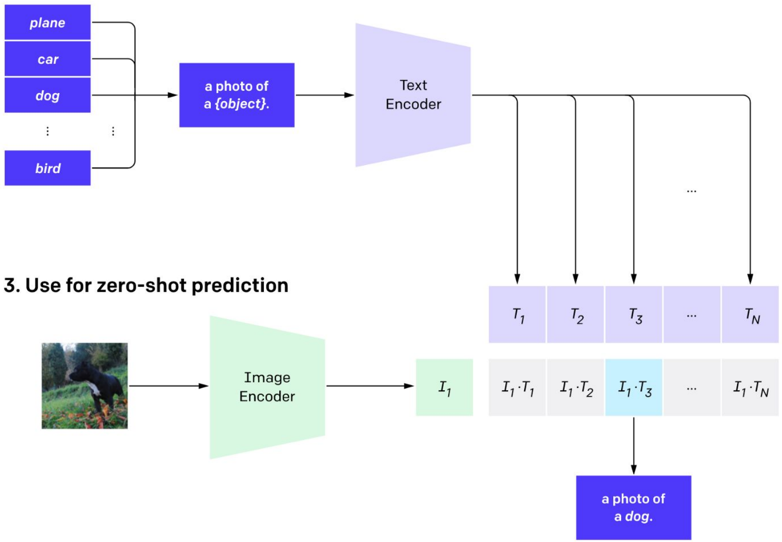
Updating Sparsity Probe

If you wish to update the code for SparsityProbe package (for instance you wish to make it work in on a new modality/ different types of inputs) - this is how to update the package

```
cd /home/ubuntu/projects/SparsityProbe
python SparsityProbe/setup.py bdist_wheel > out.txt
pip install dist/SparsityProbe-1.0-py3-none-any.whl --force-reinstall
```

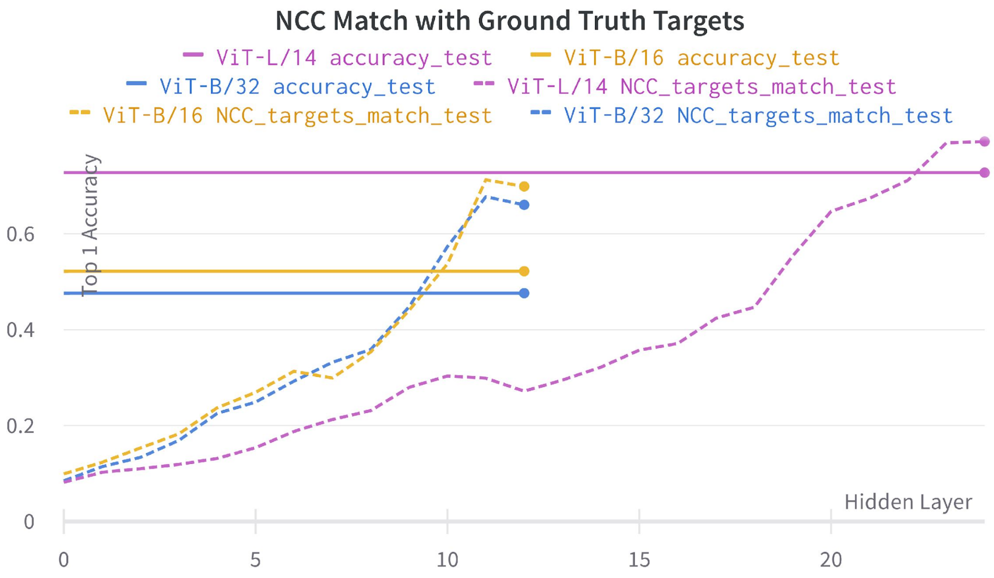
Interesting Directions - CLIP NCC Ground Truth Match

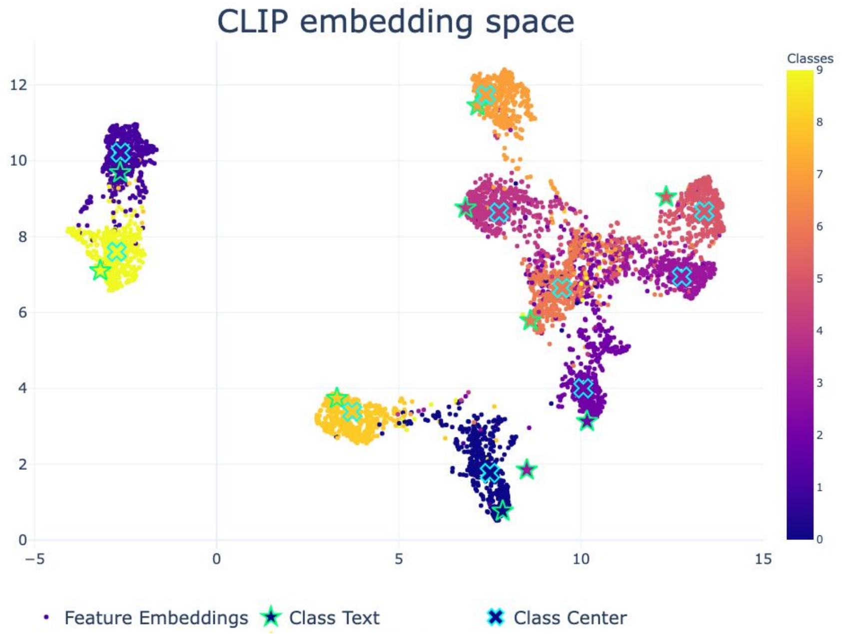




Radford, Alec et al. “Learning Transferable Visual Models From Natural Language Supervision.” ICML (2021).

Interesting Directions - CLIP NCC Ground Truth Match





Radford, Alec et al. "Learning Transferable Visual Models From Natural Language Supervision." ICML (2021).

Conclusion

"Equations are just the boring part of mathematics. I attempt to see things in terms of geometry."

Stephen Hawking

Contact:

- idobenshaul@mail.tau.ac.il

TEL AVIV UNIVERSITY