CSE 234
Data Systems for Machine Learning

Arun Kumar

Topic 3: Feature Engineering and Model Selection Systems

DL book; Chapters 8.2 and 8.3 of MLSys book
Model Selection in the Lifecycle

Data acquisition
Data preparation
Feature Engineering
Training & Inference
Model Selection
Serving
Monitoring
ML/AI + Data Systems Infrastructure

Data Scientist/ML Engineer
Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
   ❖ Feature Engineering
   ❖ Hyperparameter Tuning
   ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
   ❖ Feature Engineering Systems
❖ More Advanced Systems Issues
Bias-Variance-Noise Decomposition

ML (Test) Error = Bias + Variance + Bayes Noise

Complexity of model/hypothesis space

Discriminability of examples

Discuss the examples:
- \( x = (a, b, c); y = +1 \)
- \( x = (a, b, c); y = -1 \)

Diagram:
- High Bias
- High Variance
- Validation Error
- Training Error
Hypothesis Space of Functions

- A trained ML model is a parametric prediction function:

\[ f : \mathcal{D}_W \times \mathcal{D}_X \rightarrow \mathcal{D}_Y \]

- **Hypothesis Space:** The set \( \mathcal{H} \) of all possible functions \( f \) that can be represented by a model

- **Training:** Picks one \( f \) from hypo. space; needs estimation procedure (e.g. optimization, greedy, etc.)

- Factors that determine hypo. space:
  - Feature representation
  - Inductive bias of model
  - Regularization
Another View of Bias-Variance

- Bias arise because hypo. space does not hold “truth”
  - Shrinking hypo. space raises bias
- Variance arises due to finite training sample
  - Estimation approximately nears truth
  - Shrinking hypo. space lowers variance
3 Ways to Control Learning/Accuracy

❖ Reduce Bayes Noise:
  ❖ Augment with *new useful* features from appl.

❖ Reduce Bias:
  ❖ Enhance hypo. space: derive different features; more complex model
  ❖ Reduce shrinkage (less regularization)

❖ Reduce Variance:
  ❖ Shrink hypo. space: derive different features; drop features; less complex model
  ❖ Enhance shrinkage (more regularization)
The Double Descent Phenomenon

- DL and some other ML families can get arbitrarily complex
  - Can “memorize” entire training set
  - Curiously, variance can drop after rising; bias goes to 0!
  - Interpolation Regime is open question in ML theory

Outline

- Recap: Bias-Variance-Noise. Decomposition
- The Model Selection Triple
  - Feature Engineering
  - Hyperparameter Tuning
  - Algorithm/Architecture Selection
- Model Selection Systems
  - Feature Engineering Systems
- More Advanced Systems Issues
Unpredictability of Model Selection

- Recall 3 ways to control ML accuracy: reduce bias, reduce variance, reduce Bayes noise
- Alas, the exact raises/drops in errors on given training task and sample are not predictable
- Need empirical comparisons of configurations on data
- Train-validation-test splits; cross-validation procedures
The Model Selection Triple

The data scientist/AutoML procedure must steer 3 key activities to alter the Model Selection Triple (MST):

1. **Feature Engineering (FE):** What is/are the domain(s) of the hypo. space(s) to consider?

2. **Algorithm/Architecture Selection (AS):** What exact hypo. space to use (model type/ANN architecture)?

3. **Hyper-parameter Tuning (HT):** How to configure hypo. space shrinkage and estimation procedure approx.?
The Model Selection Triple

- The data scientist/AutoML procedure must steer 3 key activities to explore the Model Selection Triple (MST)

    - FE1
    - FE2
    - ... (FE n)
    - AS1
    - AS2
    - ... (AS n)
    - HT1
    - HT2
    - ... (HT n)

    Train and test model config(s) on ML system

    Next iteration

    Post-process and consume results

- Stopping criterion is application-specific / user-specific on Pareto surface: time, cost, accuracy, tiredness (!), etc.
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
❖ Feature Engineering
❖ Hyperparameter Tuning
❖ Algorithm/Architecture Selection
❖ Model Selection Systems
❖ Feature Engineering Systems
❖ More Advanced Systems Issues
Feature Engineering

❖ Process of converting prepared data into a *feature vector representation* for ML training/inference

❖ Aka feature extraction, representation extraction, etc.

❖ Activities vary based on data type:

- Joins and Group Bys
- Feature interactions
- Subset selection
- Temporal feature extraction
- Value recoding
- Dimensionality reduction
Feature Engineering

- Process of converting prepared data into a *feature vector representation* for ML training/inference
  - Aka feature extraction, representation extraction, etc.
- Activities vary based on data type:
  - **Text**:
    - Bag of words
    - N-grams
    - Parsing-based features
  - **Signal processing-based features**
  - Embeddings
  - Deep learning
  - Transfer learning
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
  ❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
  ❖ Feature Engineering Systems
❖ More Advanced Systems Issues
Hyperparameter Tuning

- Most ML models have hyper-parameter knobs
  - Learning rate
  - Regularization
  - Dropout prob.
  - Complexity
  - Number of trees
  - Max height/min split
  - Learning rate?

- Most of them raise bias slightly but reduce variance more
- No hyp.par. settings universally best for all tasks/data
Hyperparameter Tuning

- Common methods to tune hyperparameter configs:
  - Grid search
  - “Random” search

Hyperband

- An automated ML (AutoML) procedure for tuning hyp.par.

**Basic Idea:** For iterative procedures (e.g., SGD), stop non-promising hyp.par. configs at earlier epochs
  - Based on multi-armed bandit idea from gambling/RL

**Benefits:**
- Reapportioning resources with *early stopping* may help reach better overall accuracy sooner
- Total resource use may be lower vs grid/random search

- 2 knobs as input:
  - R: Max budget per config (e.g., # SGD epochs)
  - \( \eta \): Stop rate for configs

Hyperband

**Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.**

```plaintext
input : R, η (default η = 3)
initialization : s_{max} = \lfloor \log_\eta(R) \rfloor, B = (s_{max} + 1)R

for s ∈ \{s_{max}, s_{max} - 1, \ldots, 0\} do
    n = \left\lfloor \frac{B}{R(s+1)} \right\rfloor, \quad r = R \eta^{-s}
    // begin SUCCESSIVEHALVING with (n,r) inner loop
    T = get_hyperparameter_configuration(n)
    for i ∈ \{0, \ldots, s\} do
        n_i = \lfloor n \eta^{-i} \rfloor
        r_i = r \eta^i
        L = \{run \_then \_return \_val \_loss(t, r_i) : t ∈ T\}
        T = top_k(T, L, \lfloor n_i / \eta \rfloor)
    end
end
return Configuration with the smallest intermediate loss seen so far.
```

**Brackets:**
indiependent trials

Akin to random search

**Survival of the fittest!**

Hyperband

<table>
<thead>
<tr>
<th>$i$</th>
<th>$s = 4$</th>
<th>$s = 3$</th>
<th>$s = 2$</th>
<th>$s = 1$</th>
<th>$s = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n_i$</td>
<td>$r_i$</td>
<td>$n_i$</td>
<td>$r_i$</td>
<td>$n_i$</td>
</tr>
<tr>
<td>0</td>
<td>81</td>
<td>1</td>
<td>27</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>9</td>
<td>27</td>
<td>1</td>
<td>81</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>27</td>
<td>1</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$R = 81; \ \eta = 3$

$n_i$: # hyp.par.configs run
$r_i$: # epochs per config

- Still not as popular as grid/random search; latter is simpler and easier to use (e.g., how to set $R$ and $\eta$?)

Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
  ❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
  ❖ Feature Engineering Systems
❖ More Advanced Systems Issues
Peer Instruction Activity

(Switch slides)
Algorithm Selection

❖ **Basic Goal:** AutoML procedure to pick among a set of interchangeable models (hyp.par. tuning included)

❖ Automate a data scientist’s intuition on feature preprocessing, missing values, hyp.par. tuning, etc.

❖ Many heuristics: AutoWeka, AutoSKLearn, DataRobot, etc.

[Diagram of AutoWeka]

Algorithm Selection

- AutoScikitLearn uses a more sequential Bayesian optimization approach

NAS and AutoKeras

- DL NCG arch. akin to model family in classical ML
- Some AutoML tools aim to automate NCG design too

Google’ NAS uses RL to construct and evaluate NCGs

AutoKeras uses Bayesian optimization and has optimized impl.

- Not that popular in practice; compute-intensive; hard to debug

Review Questions

1. How does regularization affect the B-V-N tradeoff?
2. How does a key-key join affect the B-V-N tradeoff?
3. Name 2 ways to reduce bias in DL model selection.
4. Why do you think grid/random searches remain such popular forms of hyper-parameter tuning?
5. Why does Hyperband kill some configs?
6. Explain 1 pro and 1 con of automated vs manual neural architecture tuning/search.
Outline

- Recap: Bias-Variance-Noise. Decomposition
- The Model Selection Triple
  - Feature Engineering
  - Hyperparameter Tuning
  - Algorithm/Architecture Selection
- Model Selection Systems
  - Feature Engineering Systems
- More Advanced Systems Issues
Systems Aspects of Model Selection

- ML/data mining folks have studied model selection from an algorithmic automation/accuracy standpoint
- But its **resource efficiency** is a pressing ML systems issue:
  - Long running times; need lots of CPUs/GPUs
  - Cost and energy footprints non-trivial
  - If user is in the loop, latency matters too
- Need to raise **throughput** of exploring training configs with minimal resource expenses
Asynchronous Successive Halving (ASHA)

- Successor to Hyperband that uses resource more fully

**Issues -> New Ideas:**

- Top-k evals in Hyperband are sync. point bottleneck when configs are diverse -> *Asynchronous top-k check*; better for diverse configs

- Fewer and fewer configs towards bracket end (lower deg. of par.) -> *Add new hyp.par. configs* on the fly; keep all workers busy

- ASHA adapts AutoML procedure to cluster setting for massive parallel hyp.par tuning

[https://blog.ml.cmu.edu/2018/12/12/massively-parallel-hyperparameter-optimization/](https://blog.ml.cmu.edu/2018/12/12/massively-parallel-hyperparameter-optimization/)

Asynchronous Successive Halving (ASHA)

25 workers

500 workers
Total time of weeks!
**Key Observation:** False dichotomy of 2 main parallelism paradigms in ML for scalable training / model selection

### Task Parallelism
(Dask, Hyperband, ASHA, Vizier, etc.)

- High throughput model selection
- Best accuracy from Sequential SGD
- Low data scalability; wastes space (copy) or network (remote read)

### Data Parallelism
(RDBMS, Spark, PS, Horovod, etc.)

- High data scalability via sharding
  - BSP does not converge; mini-batch level has high communication costs
  - Low throughput overall
Q: Can we get the best of both worlds?
Cerebro’s Model Hopper Parallelism

- A new hybrid of task- and data-parallelism for SGD

Epoch 1.2 starts in parallel
Key Insight: SGD is robust to randomness of data ordering

Properties of Model Hopper Parallelism (MOP):

- All configs visit dataset in some sequential order; ensures similar accuracy as task parallelism
- Scheduler keeps all workers busy on shard; just like data parallelism
- No sync. point within an epoch of training all configs; very little idling of workers due to 1 comm. step per epoch
MOP is the first known form of Bulk *Asynchronous* Parallelism.

Task-Parallel Systems:
- *Dask, Celery, Vizier, Spark-HyperOpt*
- No Partitioning (Full replication)

Data-Parallel Systems:
- **MOP/CEREBRO (This Work)**
- Async. Param. Server
- Spark or TF Model Averaging
- Sync. Param. Server, Horovod

Bulk (Partitions)
Fine-grained (Mini-batches)
Communication Cost Analysis of MOP

- $p$ workers; $|S|$ configs; $k$ epochs; $b$ batch size; $m$ model size

**Table 2: Communication cost analysis of MOP and other approaches.**

*Full replication.* Remote reads.

Parameters for the example: $k = 20$, $|S| = 20$, $p = 10$, $m = 1$GB, $\langle D \rangle = 1$TB, and $|D|/b = 100$K.

<table>
<thead>
<tr>
<th></th>
<th>Comm. Cost</th>
<th>Example$^\dagger$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Hopper Parallelism</td>
<td>$kmp</td>
<td>S</td>
</tr>
<tr>
<td>Task Parallelism (FR*)</td>
<td>$p\langle D \rangle + m</td>
<td>S</td>
</tr>
<tr>
<td>Task Parallelism (RR$^\dagger$)</td>
<td>$k</td>
<td>S</td>
</tr>
<tr>
<td>Bulk Synchronous Parallelism</td>
<td>$2kmp</td>
<td>S</td>
</tr>
<tr>
<td>Centralized Fine-grained</td>
<td>$2kmp</td>
<td>S</td>
</tr>
<tr>
<td>Decentralized Fine-grained</td>
<td>$kmp</td>
<td>S</td>
</tr>
</tbody>
</table>

$2km(p - 1)|S|\left\lceil\frac{|D|}{(bp)}\right\rceil$ 72 PB
Empirical Results

- Cerebro/MOP is near Pareto-optimal on completion time, memory/space efficiency, and network cost

<table>
<thead>
<tr>
<th>System</th>
<th>ImageNet</th>
<th>GPU Utili.</th>
<th>Storage Footprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF PS - Async</td>
<td>19.00</td>
<td>8.6</td>
<td>250</td>
</tr>
<tr>
<td>Horovod</td>
<td>5.42</td>
<td>92.1</td>
<td>250</td>
</tr>
<tr>
<td>TF Model Averaging</td>
<td>1.97</td>
<td>72.1</td>
<td>250</td>
</tr>
<tr>
<td>Celery</td>
<td>1.72</td>
<td>82.4</td>
<td>2000</td>
</tr>
<tr>
<td>Cerebro</td>
<td>1.77</td>
<td>79.8</td>
<td>250</td>
</tr>
</tbody>
</table>

Figure 10: Reading data from remote storage.
Peer Instruction Activity

(Switch slides)
Vision of Cerebro Platform

Transfer Learning
Hyperparameter Tuning

Ablation Analysis
Architecture Search

Sequence Analysis
Feature Transfer

Grouped Learning
Multi-task Batching

Optimization and Scheduling Layer

Execution and Storage Layer

High-level Model Building APIs

CLIs
Jupyter

GUIs
TensorBoard
mlflow

https://adalabucsd.github.io/cerebro.html
Human-in-the-loop GUI for Cerebro

- Enables *intermittent* human-in-the-loop execution of configs
- Bridges gap between fully automated heuristics and interactive manual search

New GUI for Cerebro

- Enables *intermittent* human-in-the-loop execution of configs

**Intermittent Human-in-the-Loop Cerebro**

Upload a model script file OR choose a popular model

- Drag and Drop Select a File
- Select a popular model: 
  - ResNet-50
- Select this model

**Setup experiment**
- Name of experiment (string type)
- Description (optional)
- Name of features (string type, comma separated)
- Name of label (string type)
- Maximum training epochs (integer type)
- Data store prefix path (string type)
- Estimator function name (<module_name>::<function_name>)
- Hyperparameters search strategy (please select)

**Hyperparameter grid table:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Hyperparameter Type</th>
<th>Choices</th>
<th>Min Value</th>
<th>Max Value</th>
<th>Quantum Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning_rate</td>
<td>quantum log uniform</td>
<td>-6,-6,...,2</td>
<td>float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>batch_size</td>
<td>categorical</td>
<td>32,256</td>
<td>integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lr</td>
<td>quantum log uniform</td>
<td>-6,-6,...,2</td>
<td>float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>categorical</td>
<td>ResNet50, VGG16</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Clone Model**

- Base Model Name: Michael

**Models**

- Input model name and operate on the
  - Model Name
  - Resume Delete Stop Clone

**Experiment**

- Model Name: Michael ResNet-Example
- Christopher ResNet-Example

[Link](https://adalabucsd.github.io/papers/2021_Cerebro_VLDB_Demo.pdf)
Determined AI Training Platform

Data Prep
- Data Storage and ETL
- Spark
- S3
- Apache Airflow
- Pachyderm

Model development & training
- TensorFlow
- PyTorch
- Keras
- Hyperparameter search
- NAS
- Visualization and debugging
- Batch inference
- Distributed training
- Experiment tracking
- Cluster sharing and resource management

Model Deployment
- Web services and apps
- TensorFlow Serving
- Amazon SageMaker Hosting
- MLflow Models
- Seldon

Available today
In development

https://determined.ai/
ML users often build separate models for separate groups, e.g., by country, state, and/or age group

GROUP BY for ML; “learning over groups”

**Kingpin**: Jointly optimizes data sharding and model selection scheduling AKA “Grouped Learning”

A new hybrid of task parallelism and data parallelism
Your Reviews on Cerebro

❖ (Walked through in class)
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
   ❖ Feature Engineering
   ❖ Hyperparameter Tuning
   ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
❖ More Advanced Systems Issues
❖ (Optional) Feature Engineering Systems
End-to-End AutoML

- Some tools claim to automate data preparation too, not just feat. eng. or model selection

- Unclear how *effective* they are; no public benchmarks
- Unclear if they do any *holistic optimizations*, e.g., caching common intermediates, logical-physical separation
- Open questions on systematizing and optimizing end-to-end AutoML
More Effective Architecture Selection

- Most DL users still hand craft NCG for AS
  - Analogous to manual feat. eng. in classical ML
  - NAS / AutoKeras still have only limited adoption

- Open questions on bridging usability gap
  - Need fast human-in-the-loop tools
  - Domain-specific GUI-based AS tools?

---

https://www.youtube.com/watch?v=r5aEkpEkDzl&feature=emb_title
Cloud-Native Model Selection

- ML resource availability is now flexible and heterogenous
  - Local machine -> on-premise cluster -> cloud
- *Cloud-native* offers new opportunities/challenges:
  - **Elasticity**: upscale/downscale compute/RAM as needed
  - **Cheap decoupled storage** (e.g., S3)
  - **Cheap ephemeral compute** (e.g., Spot, Serverless)
- Redesign model selection heuristics also to cloud-native?
  - Open questions on optimizing accuracy tradeoffs vs. total cost vs. runtime
1. Name 3 model sel. systems/approaches for SGD-based ML discussed in class whose communication complexity is independent of SGD batch size.

2. Does Cerebro affect SGD convergence rates? If yes, how exactly? If not, why not?

3. In what way does ASHA improve upon Hyperband?

4. Is it possible to combine ASHA and MOP? Are there any tradeoffs involved?

5. Are there scenarios where model sel. systems are an overkill over mere single-model training systems?

6. Why bother redesigning model sel. systems for the cloud?
Discussion on Model Selection Systems
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
  ❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
❖ More Advanced Systems Issues
❖ (Optional) Feature Engineering Systems
Additional Content
(Optional; not included for quiz/exam)
Feature Engineering Systems

- Received less attention than model building systems
- **Key issues they address:**
  - **Usability:** Higher level specification of feature eng. ops
  - **Efficiency:** Automated systems-level optimization
- **Challenges:**
  - Feature eng. is very *heterogeneous*; tough for one tool to capture all ops, data types, etc.
  - *Turing-complete code* rampant in feature eng.; tough for automated optimization
Feature Engineering Systems

Sample of feature engineering systems:

- Joins
- Feature interactions
- Feature selection

- Textual / signal proc. features
- Embeddings
- Deep transfer learning

Columbus

KeystoneML

Vista & Nautilus
Feature Selection in Columbus

- **Setting**: Exploratory feature subset selection for GLMs on tabular data in R (or NumPy/Pandas)
- **Goal**: Reduce compute redundancy and data access at scale
- **Approach**: An embedded domain-specific language (DSL) with “logical” ops

---

### Example program in Columbus DSL

```plaintext
1  e  = SetErrorTolerance(0.01)         # Set Error Tolerance
2  d1 = Dataset("USCensus")            # Register the dataset
3  s1 = FeatureSet("NumHouses", ...)   # Population-related features
4  11 = CorrelationX(s1, d1)           # Get mutual correlations
5  s1 = Remove(s1, "NumHouses")        # Drop the feature "NumHouses"
6  12 = CV(lsquares_loss, s1, d1, k=5) # Cross validation (least squares)
7  d2 = Select(d1, "Income >= 10000") # Focus on high-income areas
8  s2 = FeatureSet("Income", ...)      # Economic features
9  13 = CV(logit_loss, s2, d2, k=5)    # Cross validation with (logit loss)
10 s3 = Union(s1, s2)                  # Use both sets of features
11 s4 = StepAdd(logit_loss, s3, d1)    # Add in one other feature
12 Final(s4)                           # Session ends with chosen features
```

---

https://adalabucsd.github.io/papers/2014_Columbus_SIGMOD.pdf
Feature Selection in Columbus

- **Optimization techniques:**
  - Some logical ops have alternate physical ops with different runtimes; Columbus picks automatically
  - **Exact**: Batching, Subset materialization, QR decomposition
  - **Approx.**: Coreset sampling, Warm starting

---

(b) Materialization Strategies and ROPs Used by Each Strategy

<table>
<thead>
<tr>
<th>Materialization Strategies</th>
<th>Materialization ROPs</th>
<th>Execution ROPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lazy</td>
<td>N/A</td>
<td>&lt;-, %**, solve</td>
</tr>
<tr>
<td>Eager</td>
<td>&lt;-</td>
<td>%**, solve</td>
</tr>
<tr>
<td>Naïve Sampling</td>
<td>&lt;-, sample</td>
<td>%**, solve</td>
</tr>
<tr>
<td>Coreset</td>
<td>&lt;-, %**, solve, sample, *</td>
<td>%**, solve</td>
</tr>
<tr>
<td>QR</td>
<td>&lt;-, qr</td>
<td>backsolve</td>
</tr>
</tbody>
</table>

---

(a) Error Tolerance

- Coreset+QR
- Coreset
- QR
- Lazy

(b) # Features

- Coreset+QR
- Lazy
- QR
- Coreset

(c) Size of Overlapping Feature Sets

- Coreset+QR
- Lazy
- QR
- Coreset

(d) # Iterations

- Coreset+QR
- Lazy
- QR
- Coreset

(e) # Threads

- Lazy
- Coreset+QR
Feature Pipelines in KeystoneML

- Similar to Columbus but *more general*: larger set of classical ML training and feat. eng. ops on top of Spark
- Supports text and signal proc.-based image features

```
val textClassifier = Trim andThen
  LowerCase andThen
  Tokenizer andThen
  NGramsFeaturizer(1 to 2) andThen
  TermFrequency(x => 1) andThen
  (CommonSparseFeatures(1e5), data) andThen
  (LinearSolver(), data, labels)

val predictions = textClassifier(testData)
```

- **Optimizations**: Diff. distributed linear solvers at *op level*; at *full pipeline level*: materializing and caching intermediates, sampling, common sub-expression elimination

Feature Transfer in Vista

**Setting**: Pre-trained CNNs are commonly used to extract image feature repr. for multimodal analytics

**Issue**: No single layer of CNN is universally best for downstream accuracy; need to compare multiple layers
Feature Transfer in Vista

Pre-trained Deep CNN

But no single CNN layer is always best for accuracy

From a few to >100 layers

Feature Maps

Categories

Convolution + Activation
Pooling (Subsampling)
Convolution + Activation
Fully-connected (Inner Product)

Structured Data

Brand Tags Price

Image Data
Feature Transfer in Vista

❖ **Approach**: Vista casts feature transfer as a *multi-query optimization* problem and creates *materialized views*

Naive prior approach:

Vista’s multi-query optimization:

❖ **Optimizations**: Staging out layer *materializations* avoids compute redundancy; automated memory management
Nautilus: General Transfer Learning

- Generalization of Vista to arbitrary NCGs
- **Optimizations**: *Materializations* to avoid redundancy across both models and evolving data + *model fusion* to reduce memory stalls from GPU

Tradeoffs of Feature Eng. Systems

❖ **Pros:**
  ❖ High level ops may help improve ML user productivity
  ❖ Automated resource optimization reduces costs

❖ **Cons:**
  ❖ Lack of sufficient generality
  ❖ ML user needs to (re)learn new APIs; may be complex
  ❖ Extra dependencies and maintenance issues

❖ Some companies now have in-house custom APIs/tools or general code/notebook orchestration for feat. eng. pipelines (not really optimized). More on “feature stores” in Topic 5.
More on the rise of “feature stores” in Topic 5