KeystoneML

Evan R. Sparks, Shivaram Venkataraman

With: Tomer Kaftan, Zongheng Yang, Mike Franklin, Ben Recht

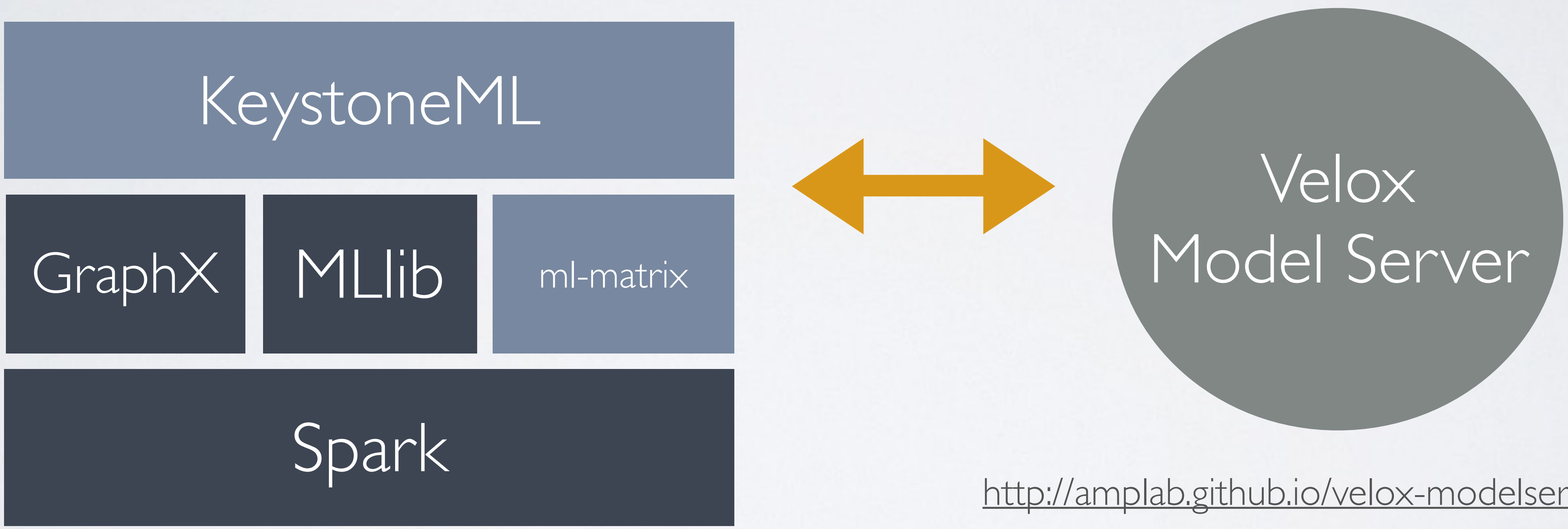
WHAT IS KEYSTONEML

- Software framework for building ***scalable end-to-end*** machine learning pipelines.
- Helps us understand what it means to build systems for ***robust***, ***scalable***, end-to-end ***advanced analytics*** workloads and the ***patterns*** that emerge.
- Example pipelines that achieve ***state-of-the-art*** results on ***large scale datasets*** in computer vision, NLP, and speech - ***fast***.
- Previewed at AMP Camp 5 and on AMPLab Blog as "ML Pipelines"
- Public release last month! http://keystone-ml.org/

HOW DOES IT FIT WITH BDAS?

Batch Model Training

Real Time Serving

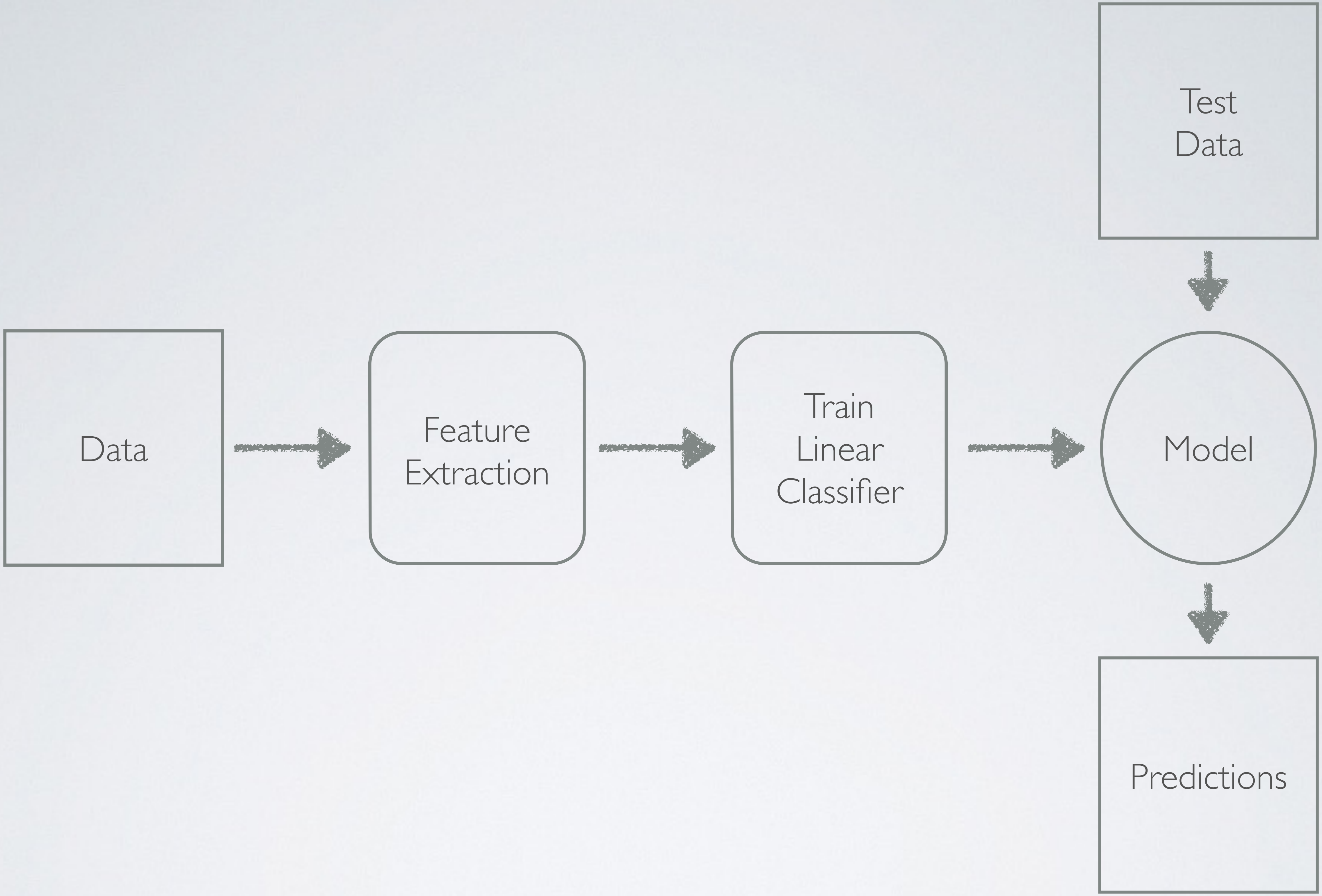


http://amplab.github.io/velox-modelserver

WHAT'S A MACHINE LEARNING PIPELINE?

A STANDARD MACHINE LEARNING PIPELINE

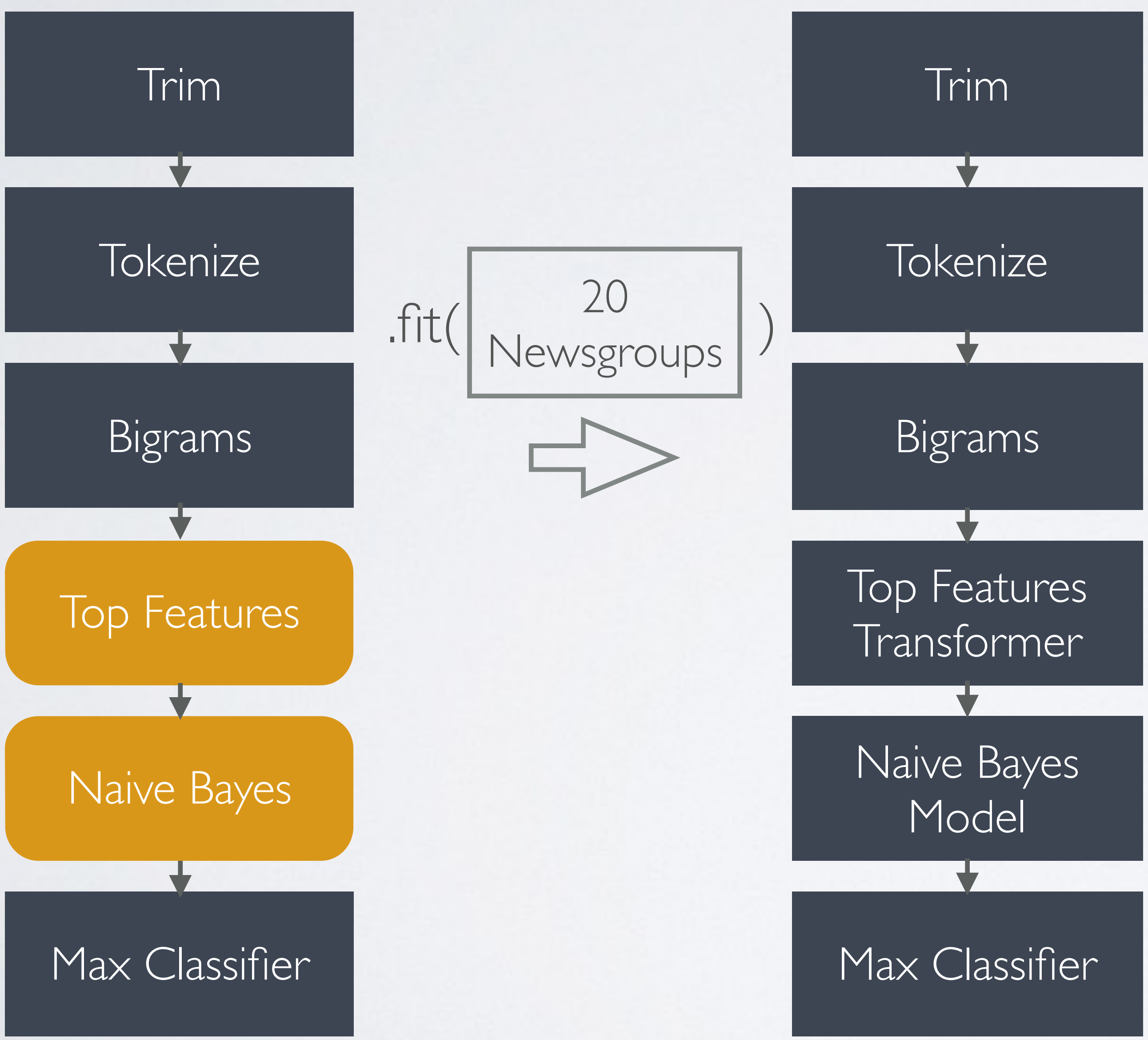
Right?



A STANDARD MACHINE LEARNING PIPELINE

That's more like it!

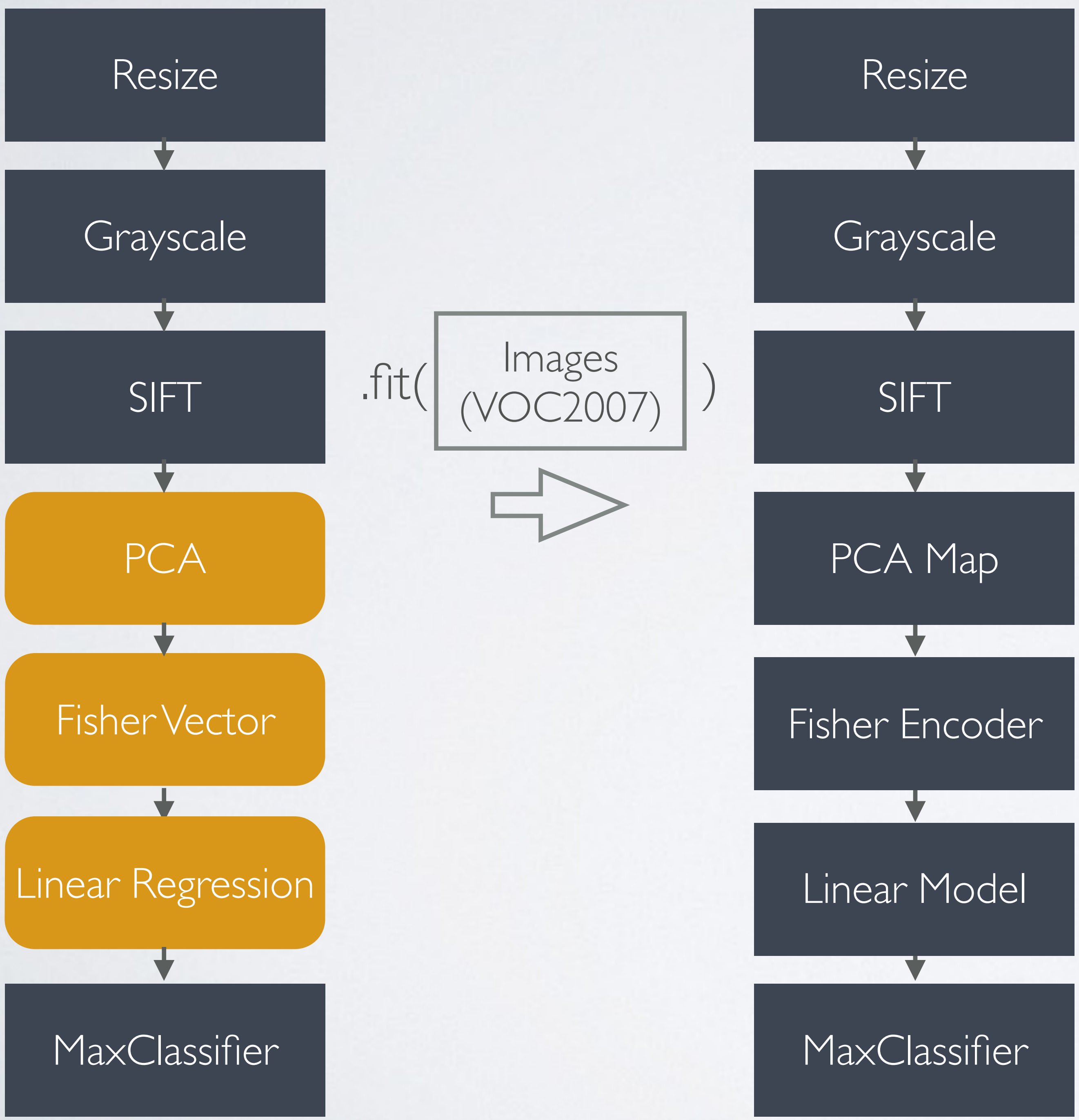
SIMPLE EXAMPLE: TEXT CLASSIFICATION




```
val predictor = Trim.then(LowerCase())
    .then(Tokenizer())
    .then(new NGramsFeaturizer(1 to conf.nGrams))
    .then(TermFrequency(x => 1))
    .thenEstimator(CommonSparseFeatures(conf.commonFeatures))
    .fit(trainData.data)
    .thenLabelEstimator(NaiveBayesEstimator(numClasses))
    .fit(trainData.data, trainData.labels)
    .then(MaxClassifier)
```

Then evaluate and ship to Velox when you're ready to apply to real time data.

NOT SO SIMPLE EXAMPLE: IMAGE CLASSIFICATION




```
val pipeline = (PixelScaler then
  GrayScaler then
  SIFTExtractor() then
  new BatchPCATransformer(pcaMat) then
  new FisherVector(gmm) then
  FloatToDouble then
  MatrixVectorizer then
  NormalizeRows then
  SignedHellingerMapper thenLabelEstimator
  new BlockLeastSquaresEstimator(blockSize, numPasses, lambda))
    .fit(trainImages, trainingLabels)
```

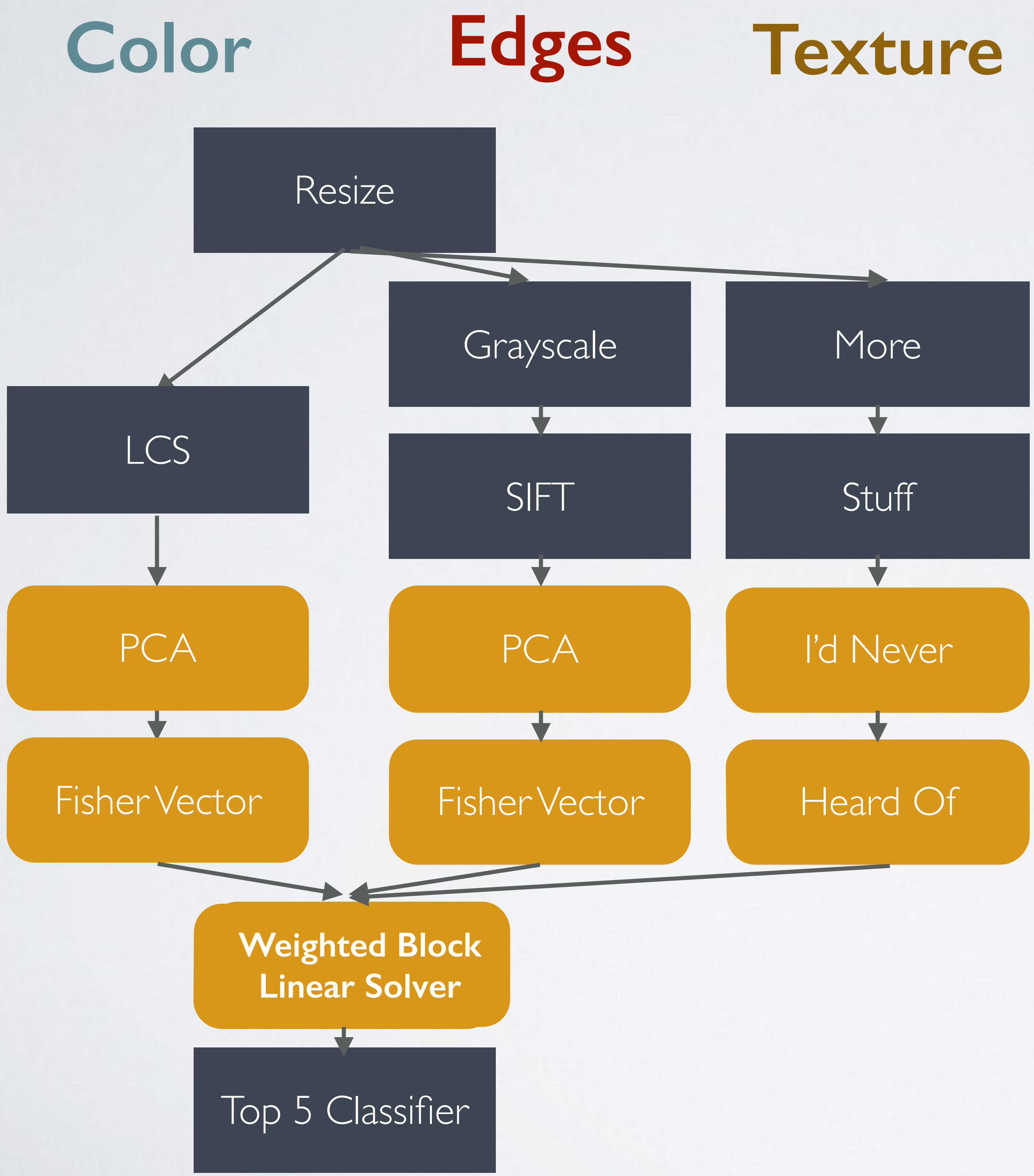
5,000 examples, 40,000 features, 20 classes

Embarassingly parallel featurization and evaluation.

15 minutes on a modest cluster.

Achieves performance of Chatfield et. al., 2011

EVEN LESS SIMPLE: IMAGENET



<200 SLOC

1000 class classification.
1,200,000 examples
64,000 features.

90 minutes on 100 nodes.

And Shivaram doesn't have a heart attack when Prof. Recht tells us he wants to try a new solver.

Or add 100,000 more texture features.

SOFTWARE FEATURES

- Data Loaders
  - CSV, CIFAR, ImageNet, VOC, TIMIT, 20 Newsgroups
- Transformers
  - NLP - Tokenization, n-gra
    parsing*
  - Images - Convolution, Gr
    FisherVector*, Pooling, W
  - Speech - MFCCs*
  - Stats - Random Features, Normalization, Scaling*, Signed Hellinger Mapping, FFT
  - Utility/misc - Caching, Top-K classifier, indicator label mapping, sparse/dense encoding transformers.
- Estimators
  - Learning - Block linear models, Linear Discriminant Analysis, PCA, ZCA Whitening, Naive Bayes*, GMM*

- Example Pipelines
  - NLP - 20 Newsgroups, Wikipedia Language model
  - T, CIFAR, VOC, ImageNet
  - Binary Classification
  - Multiclass Classification
  - Multilabel Classification

Just 5k Lines of Code,
1.5k of which are TESTS
+ 1.5k lines of JavaDoc

* - Links to external library: MLlib, ml-matrix, VLFeat, EncEval

KEY API CONCEPTS

TRANSFORMERS

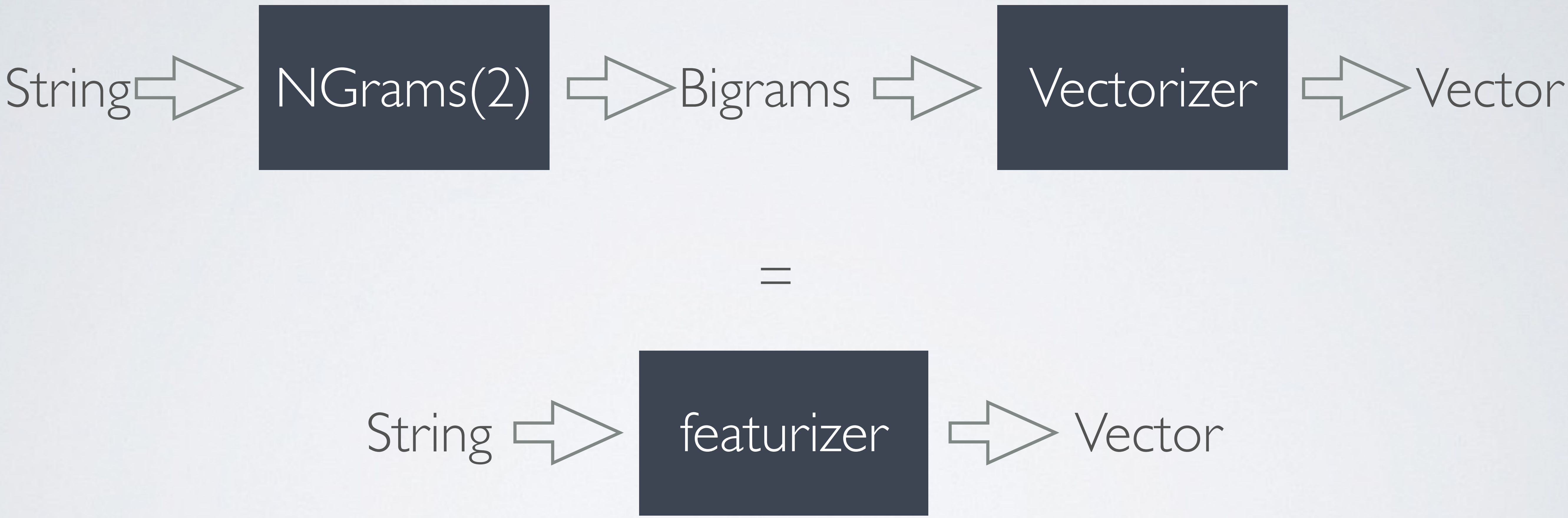

```
abstract class Transformer[In, Out] {
    def apply(in: In): Out
    def apply(in: RDD[In]): RDD[Out] = in.map(apply)
    …
}
```

TYPE SAFETY HELPS ENSURE ROBUSTNESS

ESTIMATORS

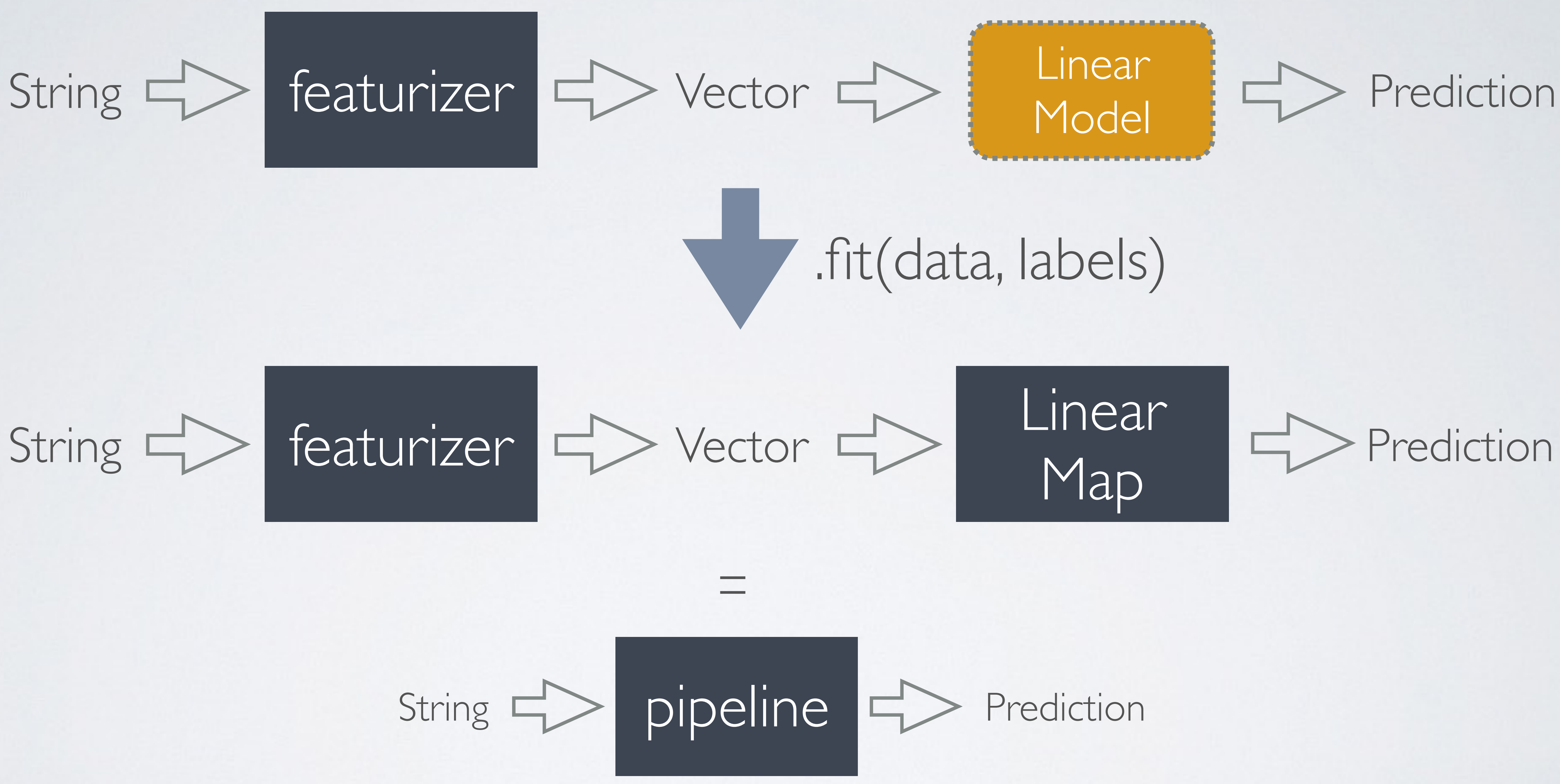

```
abstract class Estimator[In, Out] {
    def fit(in: RDD[In]):Transformer[In,Out]
    ...
}
```

CHAINING



val featurizer: Transformer[String, Vector] = NGrams(2) ***then*** Vectorizer

COMPLEX PIPELINES




```
val pipeline = (featurizer thenLabelEstimator LinearModel).fit(data, labels)
```

USING THE API

CREATE A TRANSFORMER

"I want a node that takes a vector and divides by it's two-norm."

- Can be defined as a class.

```
object Normalizer
    extends Transformer[Vector[Double], Vector[Double]] {
  def apply(x: Vector[Double]): Vector[Double] = {
    val norm = sqrt(sum(pow(x, 2.0)))
    x / norm
  }
}
```

- Or as a Unary function.

```
def normalize(x: Vector[Double]) =
  x / sqrt(sum(pow(x, 2.0)))
```

Chain with other nodes:
val pipeline = Vectorizer then Normalizer
or
val pipeline = Vectorizer thenFunction normalize _

INTEGRATING WITH C CODE?

Don't reinvent the wheel - use JNI!

Loads a shared library

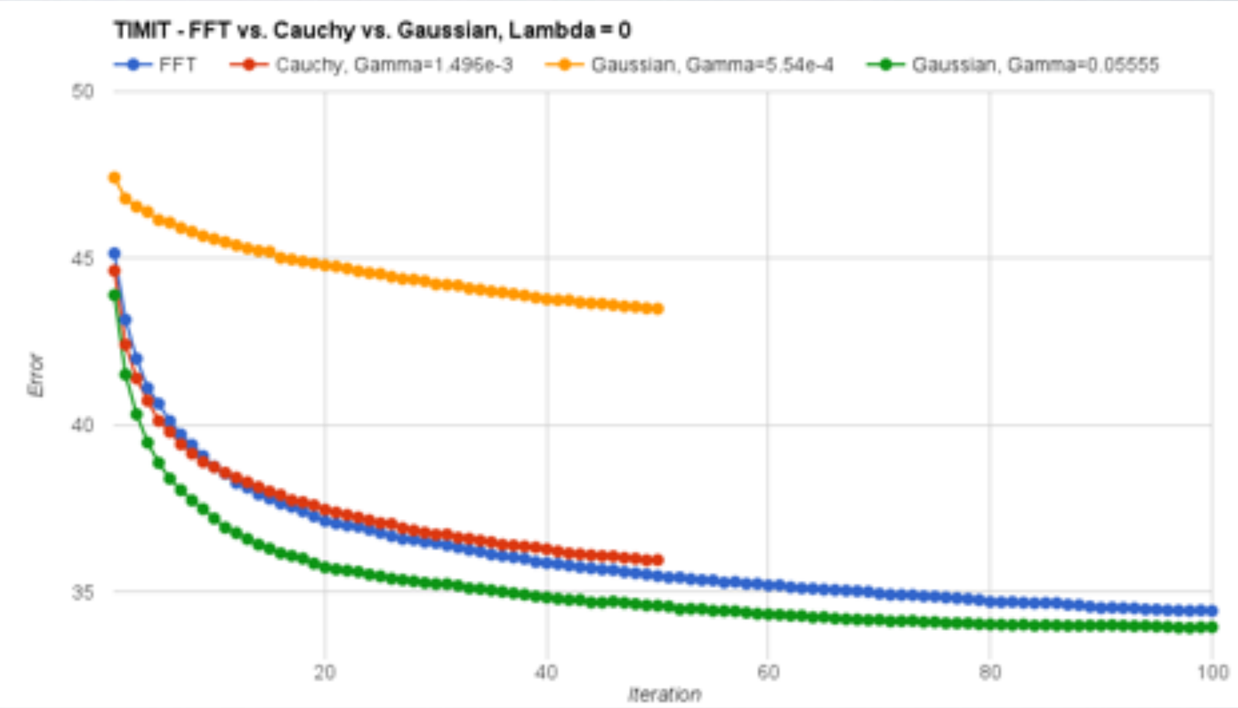
```
class VLFeat extends Serializable {
  System.loadLibrary("ImageFeatures")
  // This will load libImageEncoders.{so,dylib} from the library path.

  /**
   * Gets SIFT Descriptors at Multiple Scales emulating the
   * `vl_phow` MATLAB routine. Under the hood it uses
   * vl_dsift from the vlfeat library.
   *
   * @param width Image Width.
   * @param height Image Height.
   * @param step Step size at which to sample SIFT descriptors.
   * @param bin SIFT Descriptor bin size.
   * @param numScales Number of scales to extract at.
   * @param image Input image as float array.
   * @return SIFTs as Shorts.
   */
  @native
  def getSIFTs(
      width: Int,
      height: Int,
      step: Int,
      bin: Int,
      numScales: Int,
      scaleStep: Int,
      image: Array[Float]): Array[Short]

}
```

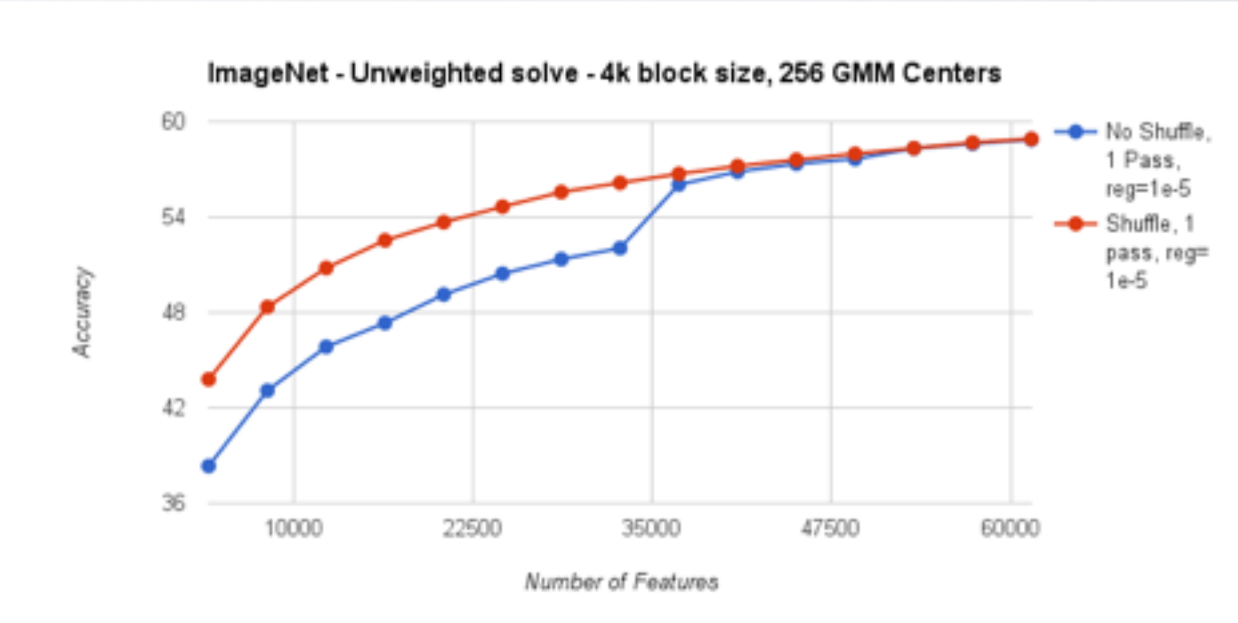
`javah` generates a header for the shared library.

Native code is shared across the cluster.

RESULTS

- TIMIT Speech Dataset:
  - Matches state-of-the-art statistical performance.
  - 90 minutes on 64 EC2 nodes.
  - Compare to 120 minutes on 256 IBM Blue gene machines.
- ImageNet:
  - 67% accuracy with weighted block coordinate decent. Matches best accuracy with 64k features in the 2012 ImageNet contest.
  - 90 minutes end-to-end on 100 nodes.





RESEARCH COMPLEMENT TO MLLIB PIPELINE API

MLlib Pipeline API

- Basic set of operators for text, numbers.
- All spark.ml operations are transformations on DataFrames.
- Scala, Java, Python
- Part of Apache Spark

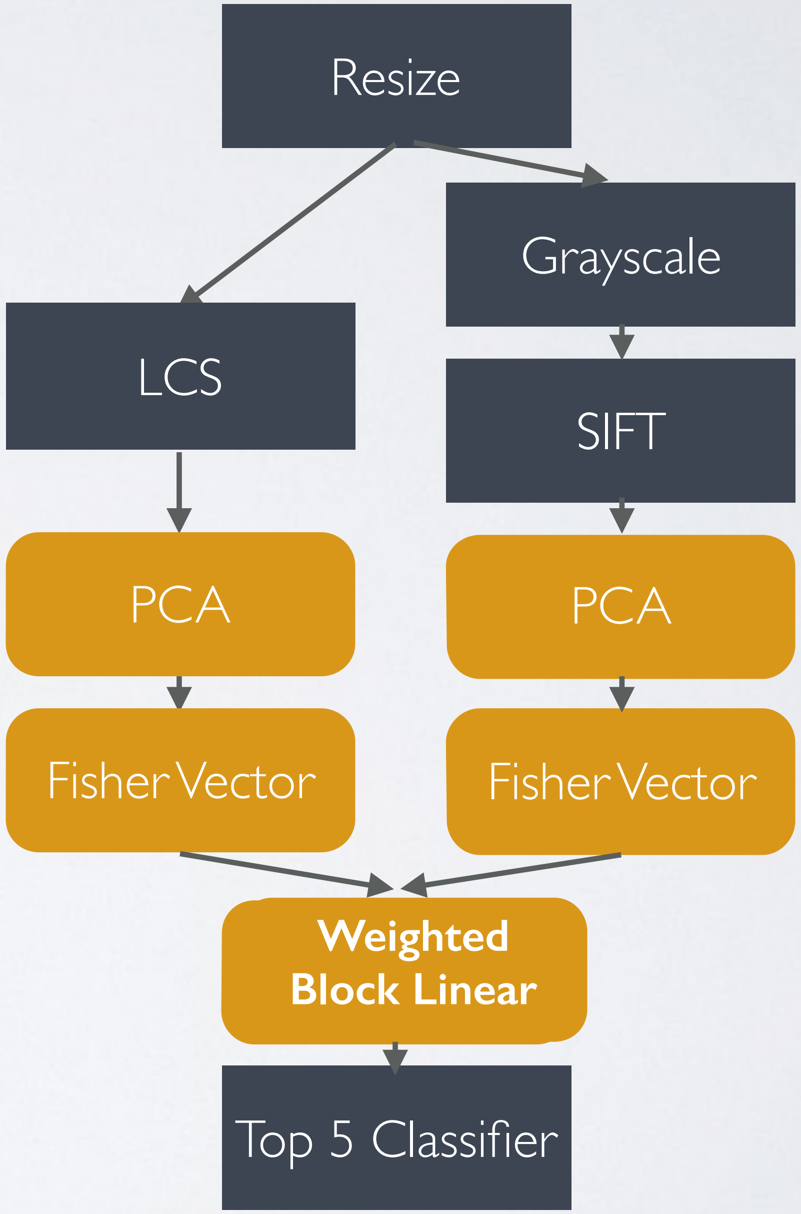
KeystoneML

- Enriched set of operators for complex domains: vision, NLP, speech, plus advanced math.
- Type safety.
- Scala-only (for now)
- Separate project (for now)
- External library integration.
- Integrated with new BDAS technologies: Velox, ml-matrix, soon Planck, TuPAQ and SampleClean

RESEARCH DIRECTIONS

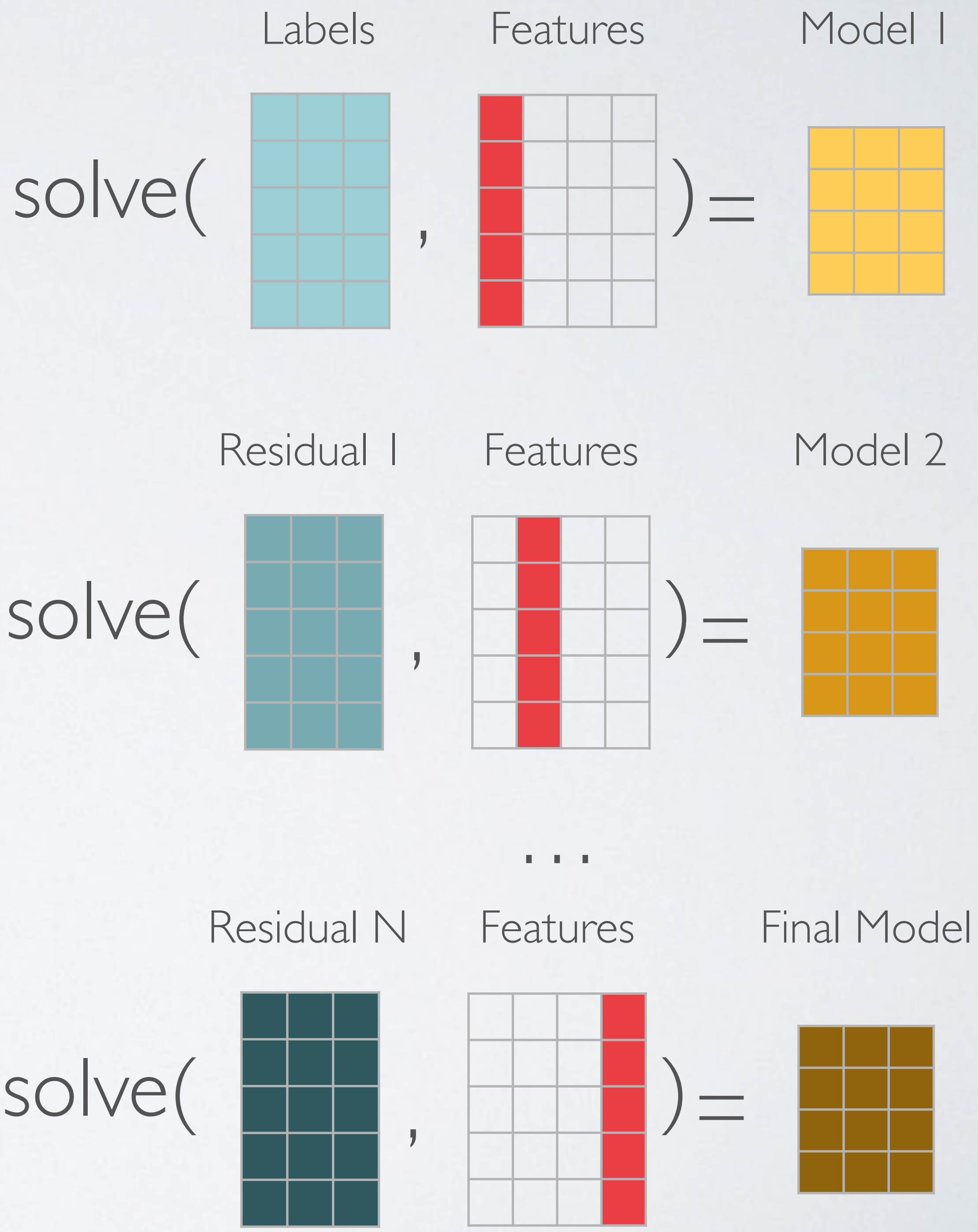
AUTOMATIC RESOURCE ESTIMATION

- Long-complicated pipelines.
  - Just a composition of dataflows!
- When how long will this thing take to run?
- When do I cache?
  - Pose as a constrained optimization problem.
- Enables Efficient Hyperparameter Tuning



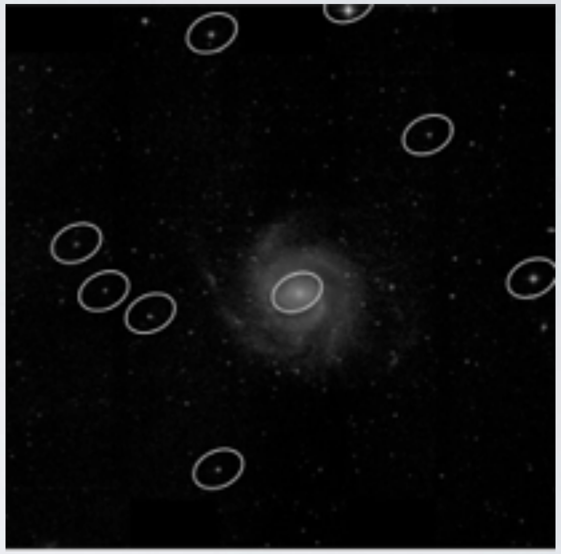
BETTER ALGORITHMS

- Linear models are pretty great.
  - Can be slow to solve exactly.
  - Need ***tons*** of RAM to materialize full, dense, feature spaces.
  - In classification - bad at class imbalance.
- We've developed a new algorithm to address all of these issues.
  - Block Weighted Linear Solve
  - Distributed, lazily materialized, weighted, and approximate.
  - Can be orders of magnitude faster than direct solve methods, and much more accurate for highly imbalanced problems.

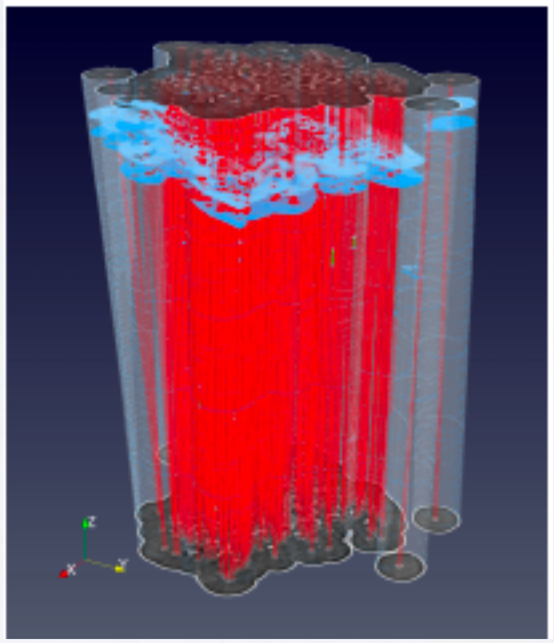


MORE APPLICATIONS

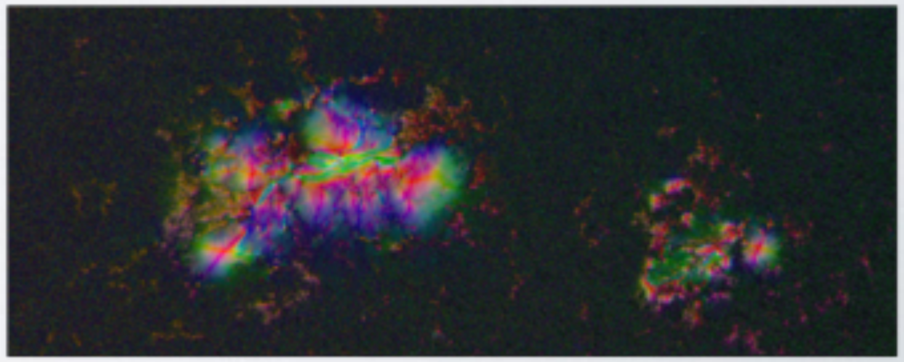
- Ongoing collaborations involving:
  - Astronomical image processing.
    - Count the stars in the sky!
    - Find sunspots with image classification.
  - High resolution structure recognition for materials science.
    - Images so big they are RDDs!
  - Advanced Language Models.
    - Scalable Kneser-Ney smoothing for high-quality machine translation.



Z Zhang, et. al, 2014



D. Ushizima, et. al, 2014



E. Jonas, et. al, 2015

QUESTIONS?

KeystoneML

http://keystone-ml.org/

Code

http://github.com/amplab/keystone

Contributors:

Daniel Bruckner, Mike Franklin, Glyfi Gudmundsson, Tomer Kaftan, Daniel Langkilde, Henry Milner, Ben Recht, Vaishaal Shankar, Evan Sparks, Shivaram Venkataraman, Zongheng Yang