e2eML: High Performance, Power Efficient Application of End-to-End Machine Learning Systems

Mauricio Breternitz
INESC-ID
University of Lisbon

Shonan Seminar: Advances on Heterogenous Computing from HW to SW

sep.03.2018

Outline

Introduction

End-to-End DNN

Potential Benefits- Global Optimization

Challenges

Data Normalization, Artifacts, Adaptation

Approach and Use Cases

Related – TVM, Weld

Research Framework & Plan

Research Support by Fundação para a Ciência e a Tecnologia (FCT) under project UID/CEC/50021/2013

Brief BIO, Publications, Patents

PhD – Carnegie-Mellon, ECE
MSc – UNICAMP/Brazil
BSc – ITA-Brazil

Work: IBM Research, Motorola, Times N, Intel Labs, AMD Research

49 U.S. Patents Issued, 54 U.S. Patents Pending

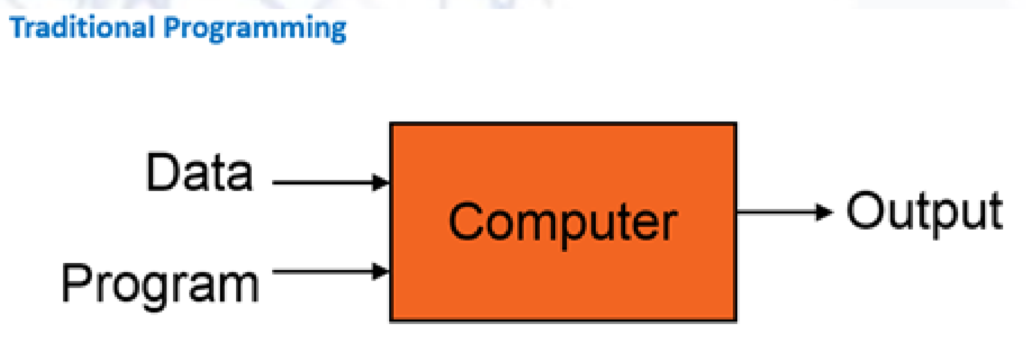
Publications

Citations 1609 H-index 24, i10-index 40
Computer Architecture, Computer Systems, Performance, Tuning
Big Data, Machine Learning

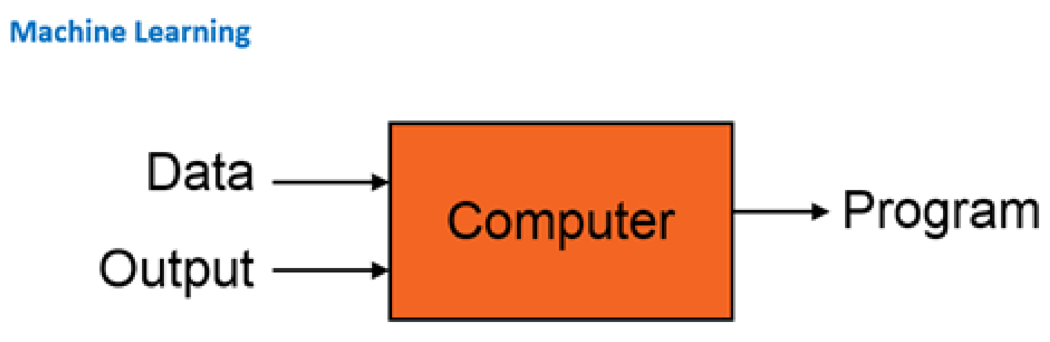
Creator /General Chair : AMAS-BT International Workshop on Architectural/Microarchitectural Support for Binary Translation, joint with ISCA and CGO.

DEFINITION OF MACHINE LEARNING

- Simple Definition: “Algorithms that Learn



In Traditional Programming, a Human expert encodes his knowledge of the relationship of data and desired output as a program to process input data to generate the desired output



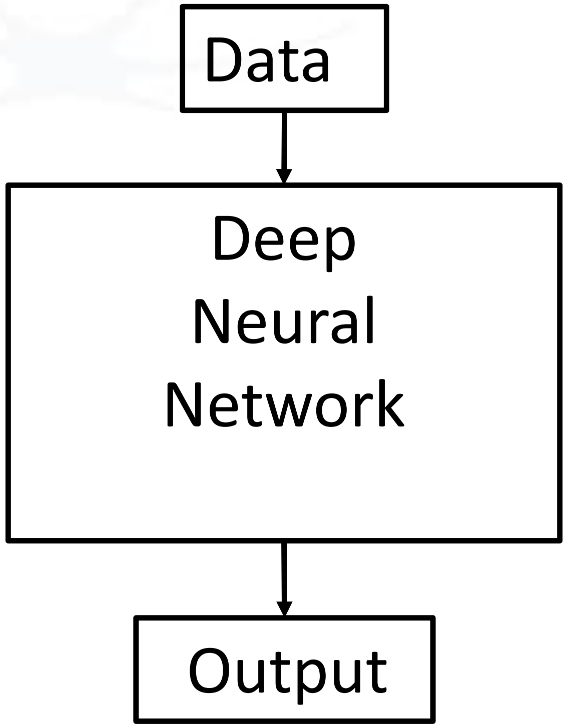
In Machine Learning, the system autonomously learns the relationship of data and the desired output, creating classification rules (inference) to provide the desired output from similar input

◢ Machine Learning: A system capable of the autonomous acquisition and integration of knowledge

Challenge: Black Box -> Hardware

End-To-End DNN

- Idealized Framework



Conjecture

- End-To-End DNN may lead to globally optimized results
  - Tailored code generator optimizing across layers may be better than hand-tuned generic libraries
  - Trained Network: tailored interconnect

MACHINE LEARNING:

APPLICATION FRAMEWORK across MULTIPLE REGIONAL SCOPES

data travel distance →

Point of Data Capture → **Local/Regional Data Center** → **Scale-Out Data Center**

Point of Data Capture	Local/Regional Data Center	Scale-Out Data Center
Compute reaction nearby the place data is captured, using locally stored knowledge	Compute reaction using aggregate data from a few, logically neighboring compute sites	Compute reaction using knowledge from a worldwide and long term memory
Examples - Car warning system using front camera image; Factory Sensor Input	Example: Traffic management and scheduling of a city-wide fleet	Example: Netflix movie recommendation engine

HW Substrate(s)

- CPU, multicores
- Accelerators: CPU+GPU, CPU+FPGA
- Cloud
  - Accelerator cloud
- Edge, IoT

End-to-End ML Challenges

-Network Inspection

-Who does what?

-Adversarial Input

HW Substrate Requirements

- Reliability
- Resiliency

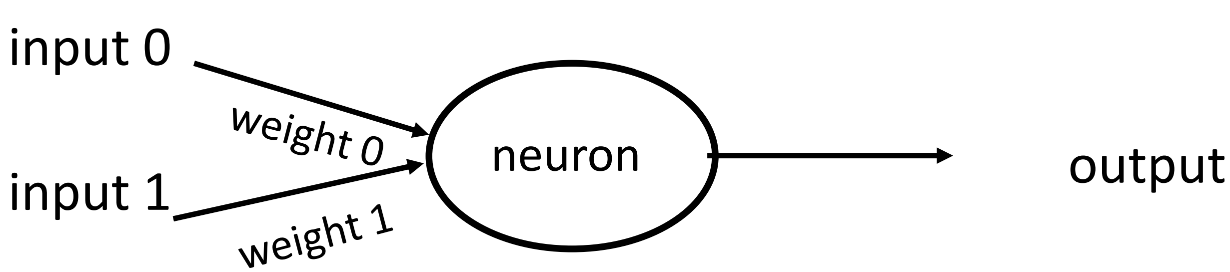
- Power Consumption
  - Absolute
  - Efficiency
- Latency
- Cost

KEY MACHINE LEARNING ALGORITHMS

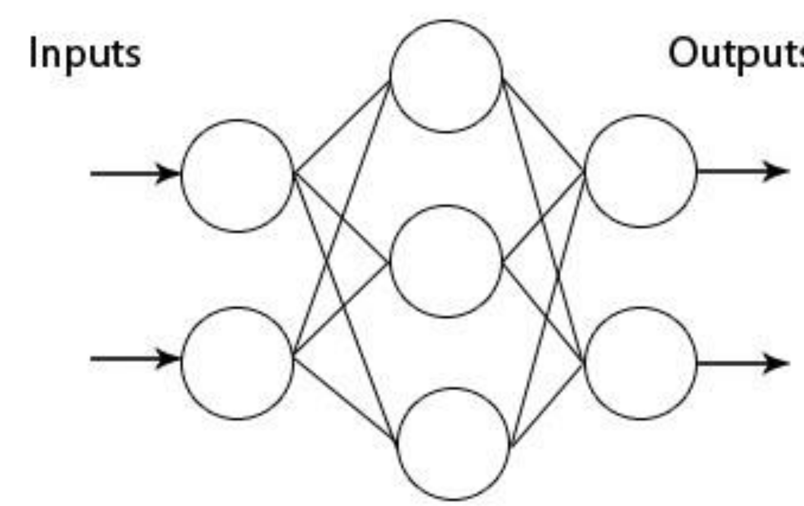
- *Classes of Machine Learning Algorithms:*
  - *Statistically-inspired algorithms: Bayesian Networks, Logistic Regression, Decision Trees, etc.*
  - *Deep Neural Networks(DNN)*
    - *rapidly becoming the preferred algorithm, currently provide the best solutions for image/speech/natural language processing*
    - *Biologically-inspired: simulated neurons*
    - *DNNs are a good match for heterogeneous (GPU, FPGA) acceleration because the mathematical operation to compute the effects of weighted inputs for multiple neurons is a matrix-vector multiplication.*

DEEP NEURAL NETWORKS

- *rapidly becoming the preferred algorithm, currently the best solutions for image/speech/natural language processing*
- *Biologically-inspired: simulated neurons*
- *Good match for AMD GPU acceleration because the mathematical operation to compute the effects of weighted inputs for multiple neurons is a matrix-vector multiplication.*

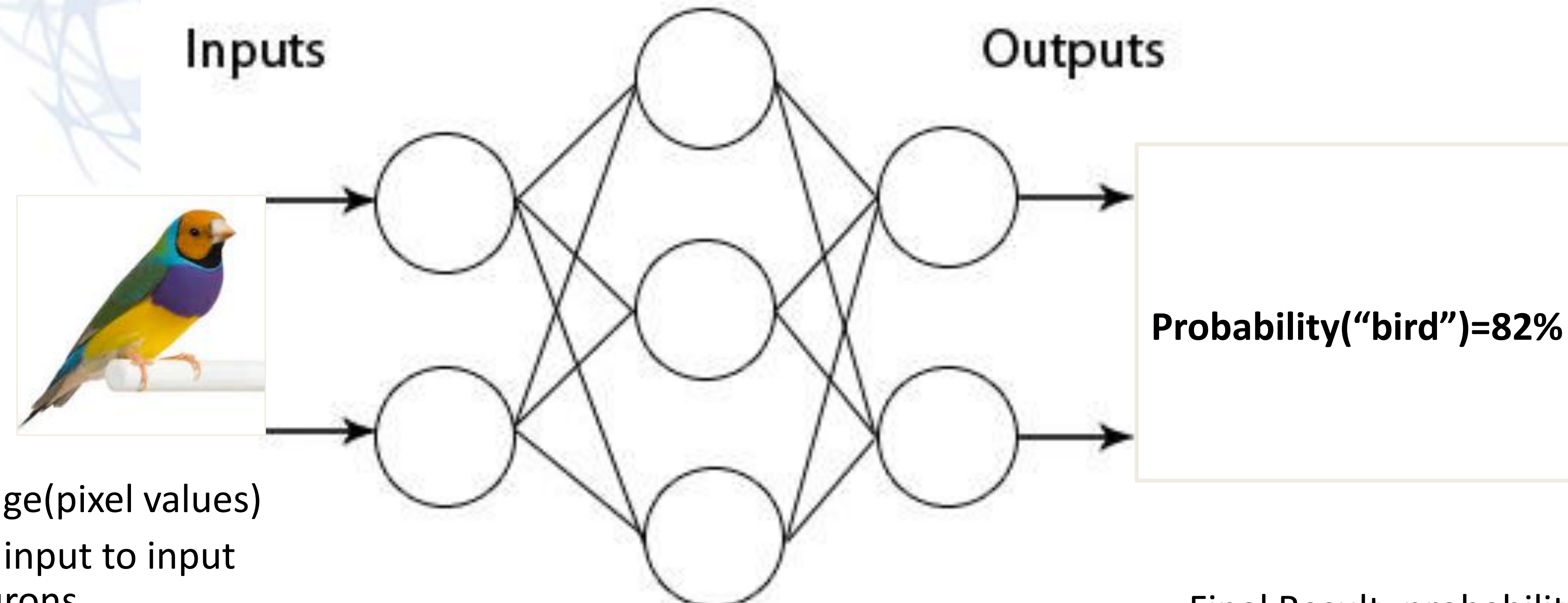


*A Simulated Neuron: A biologically inspired algorithm whereby a number of input values are provided to a simulated neuron, which computes an output based on a **weighted** combination of the input values*



An Example Deep Neural Network(DNN): A multi-layered sequence of simulated neurons

EXAMPLE: DEEP NEURAL NETWORK CLASSIFYING AN IMAGE



Image(pixel values) are input to input neurons

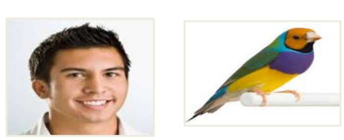
Groups of neurons are trained to detect specific features in image regions.

Subsequent layers reacts to combinations of features that compose the image.

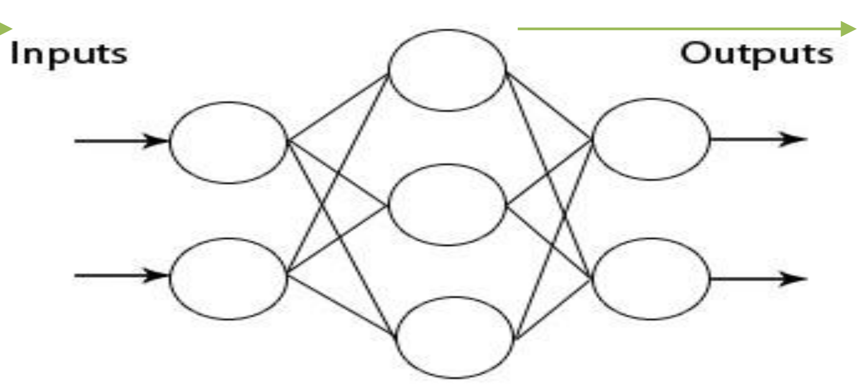
Final Result: probability of input image being a given entity

INFERENCE

- **Is** the problem of identifying to which [categories]() a new [observation]() belongs
  - Examples - Sort and Classify input into discrete categories
    - Create photo categories from input set (exemplified on previous slide)
    - Email: {message} classified as one of { spam, NOT-spam}
    - Diagnosis: { gender, symptoms} used to determine disease
  - Uses *Trained* deep networks for RECOGNITION tasks
  - Focus on efficient Forward computation
  - Focus on Latency: Minimize end-do-end response time: smaller mini-batches



a number of images



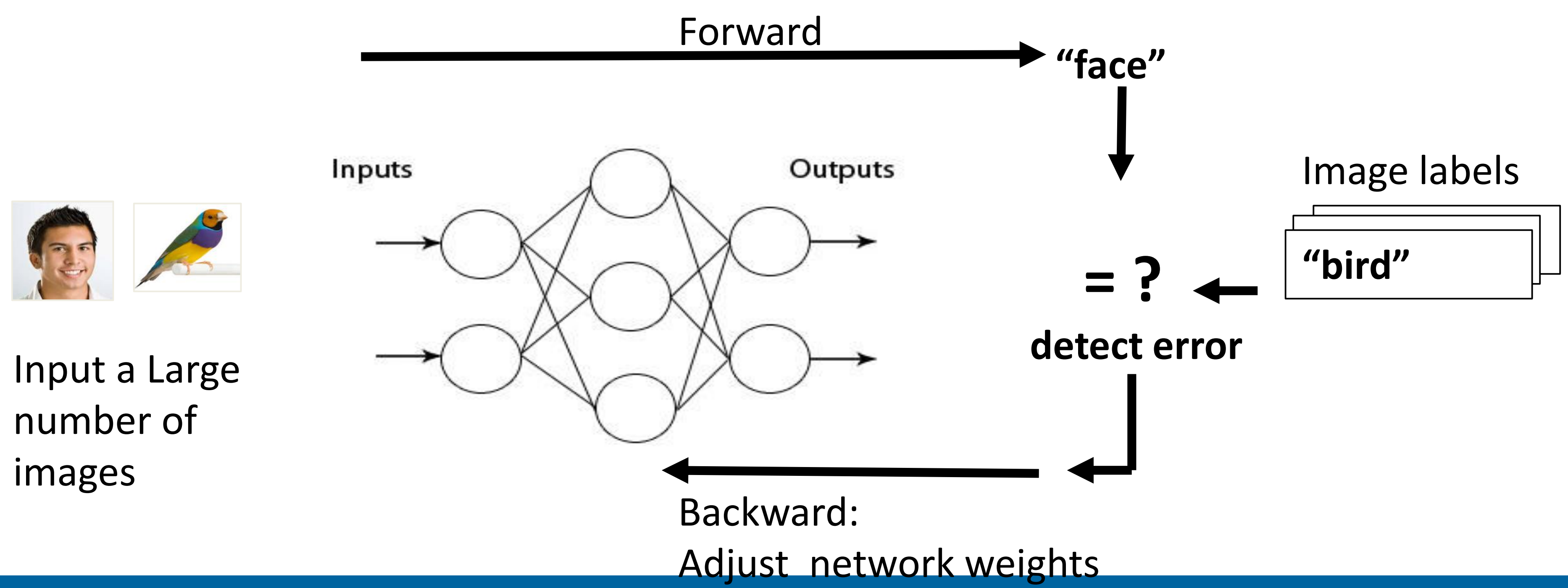
“bird”, “ face”, …

Inference: images are presented to the network to determine what class of image it is

TRAINING

- or LEARNING is the computationally-demanding task of determining the parameters of a neural network
  - Has both Forward, Backward propagation phases
  - State of the Art is GPU Acceleration
  - Focus on High Throughput

Images are presented to the network to determine class; erroneous outputs are propagated backwards correcting network parameters to achieve high accuracy:



End To End Machine Learning

Self Driving Car Example:
- Map camera pixels to steering command
- System learns internal representation
- Avoids explicit system decomposition
  - Lane marking, path planning, control

Efficiency:
- Optimized for maximal overall performance
- Enable smaller networks

Training and Inference

Data Collection

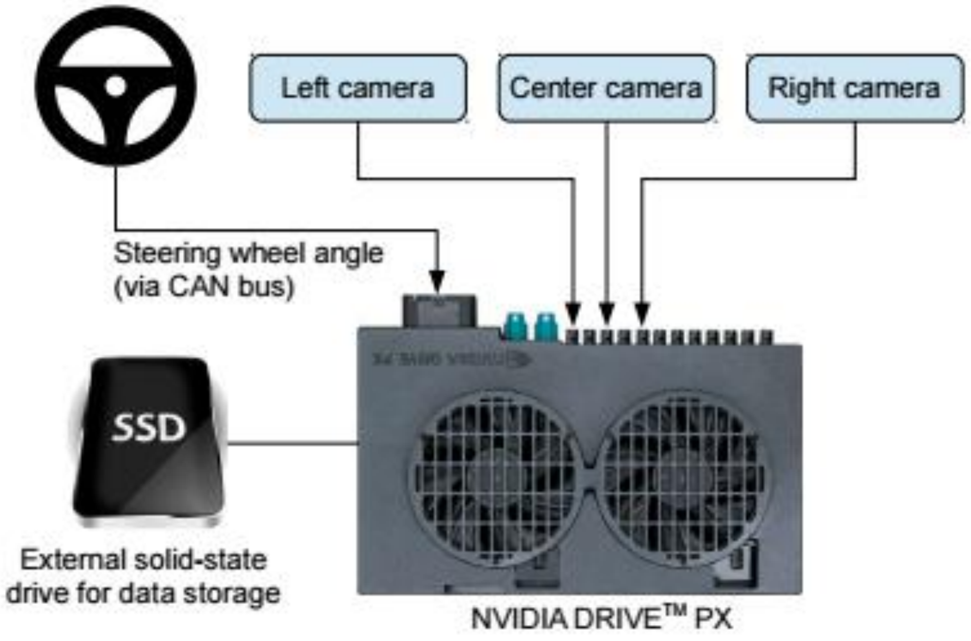
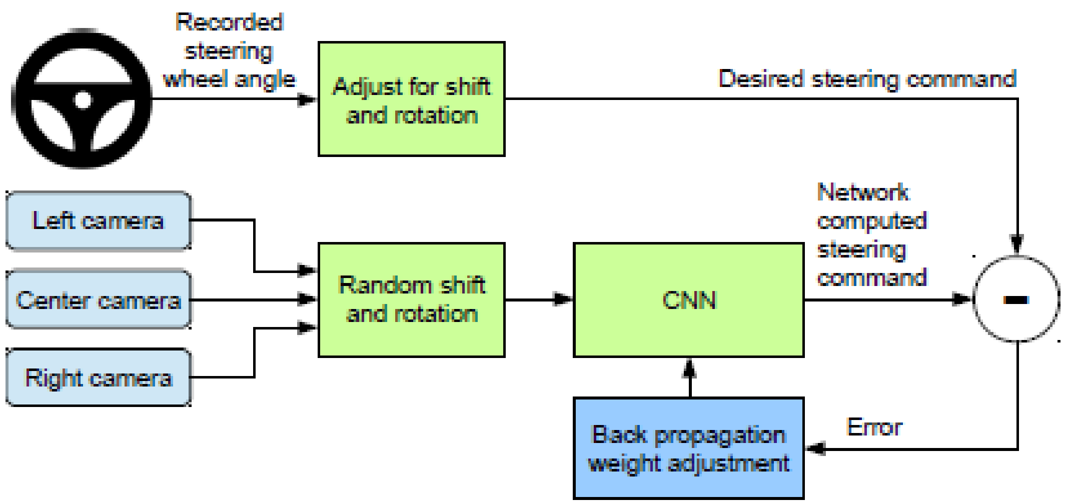


Figure 1: High-level view of the data collection system.

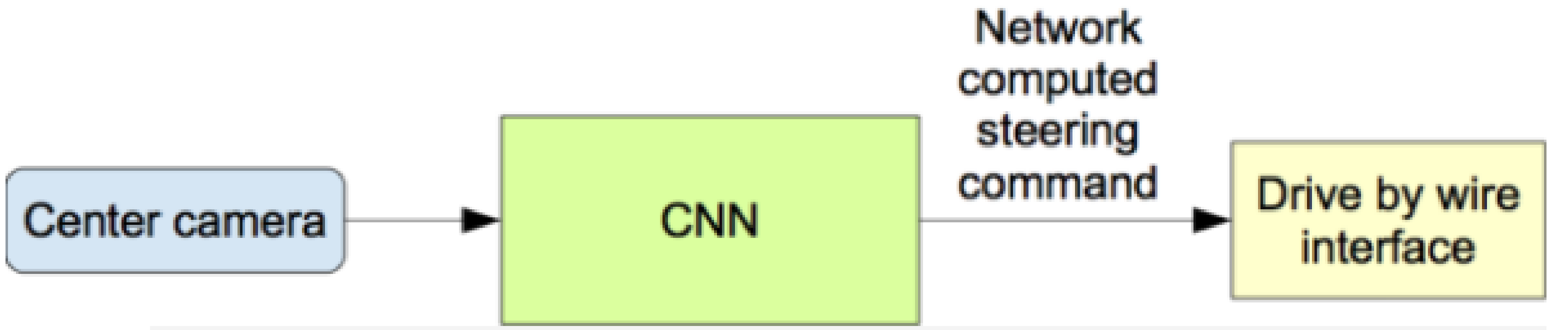
Training



Training the neural network.

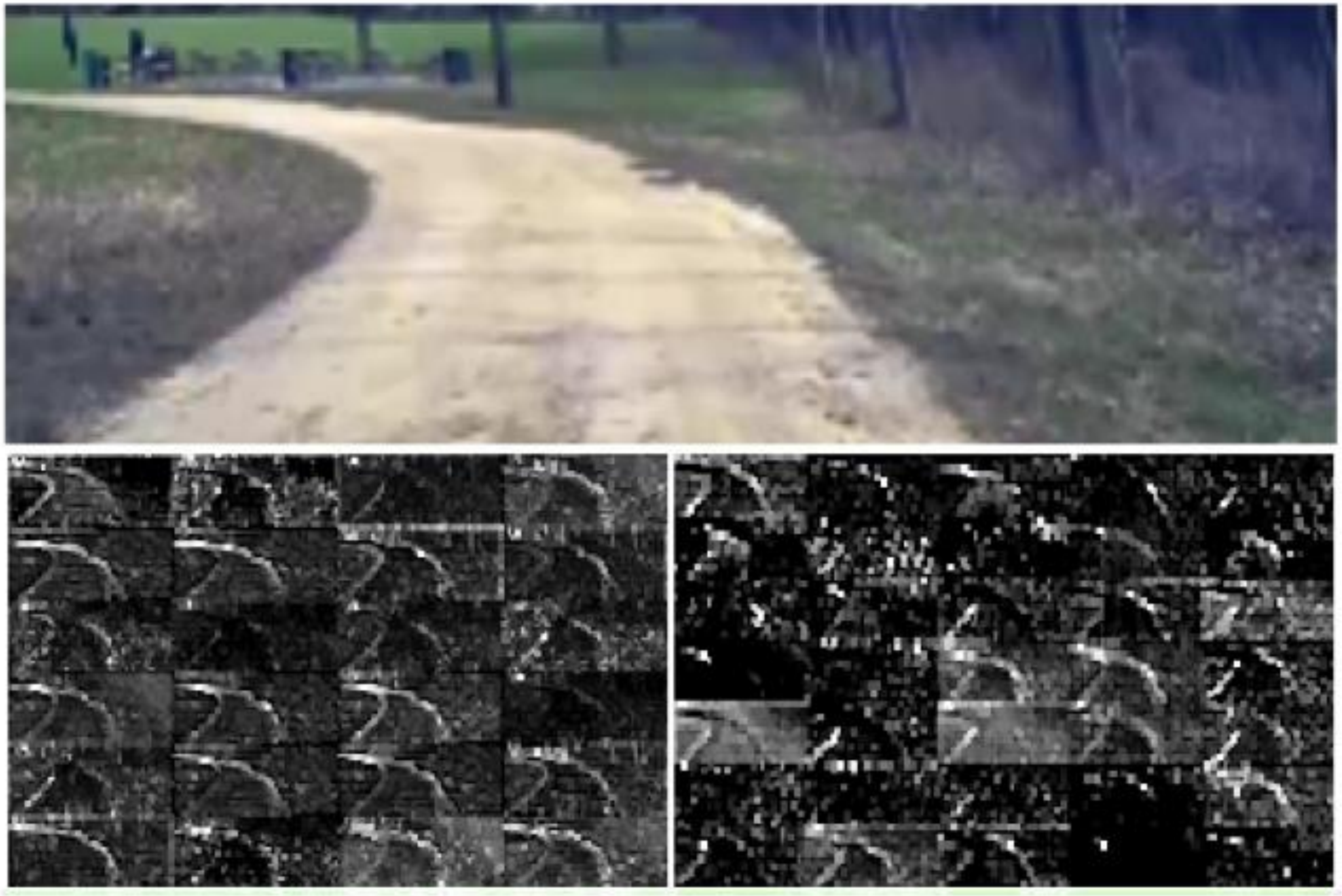
Training and Inference

Inference



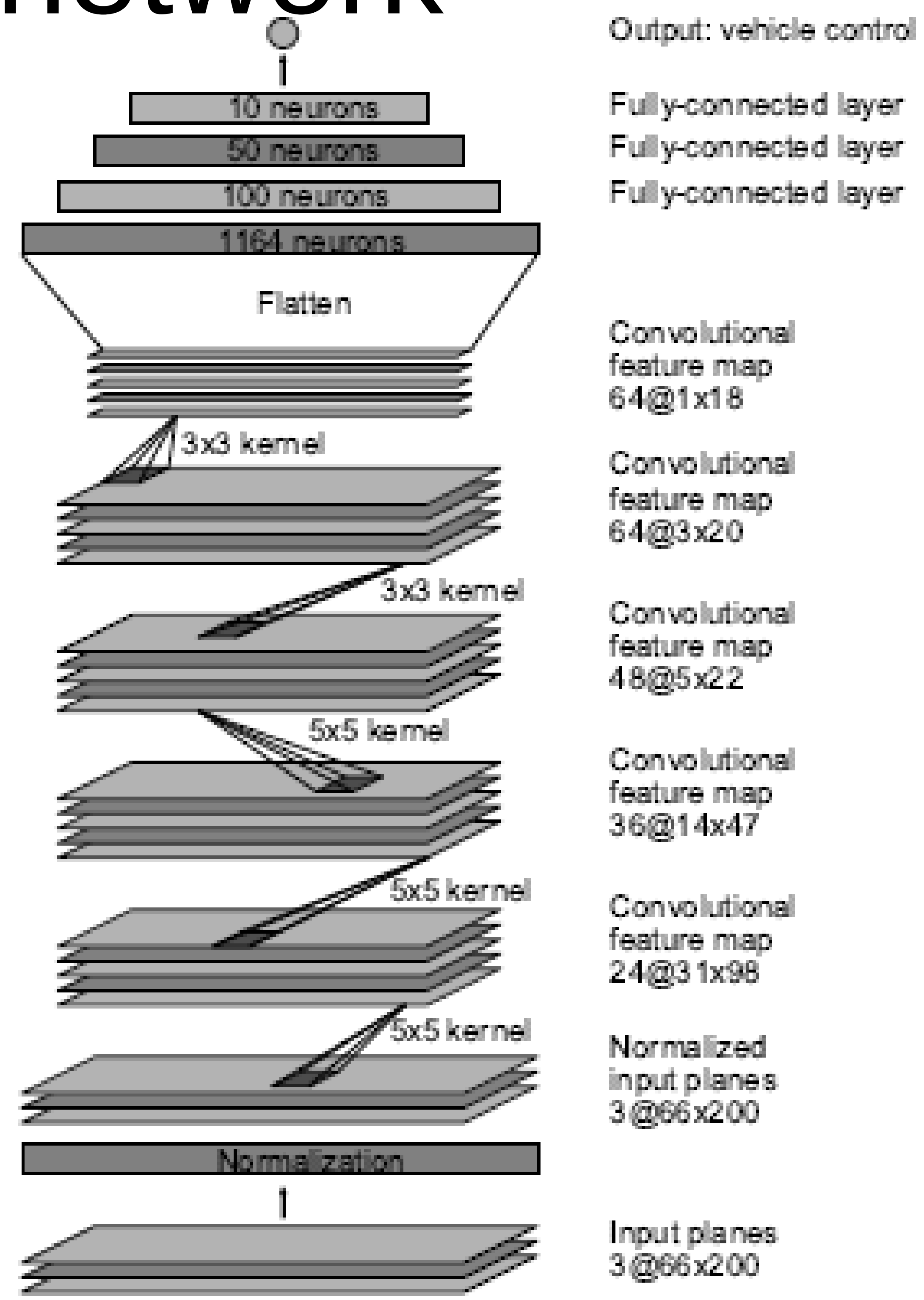
The trained network is used to generate steering commands from a single front-facing center camera.

Road Image Example



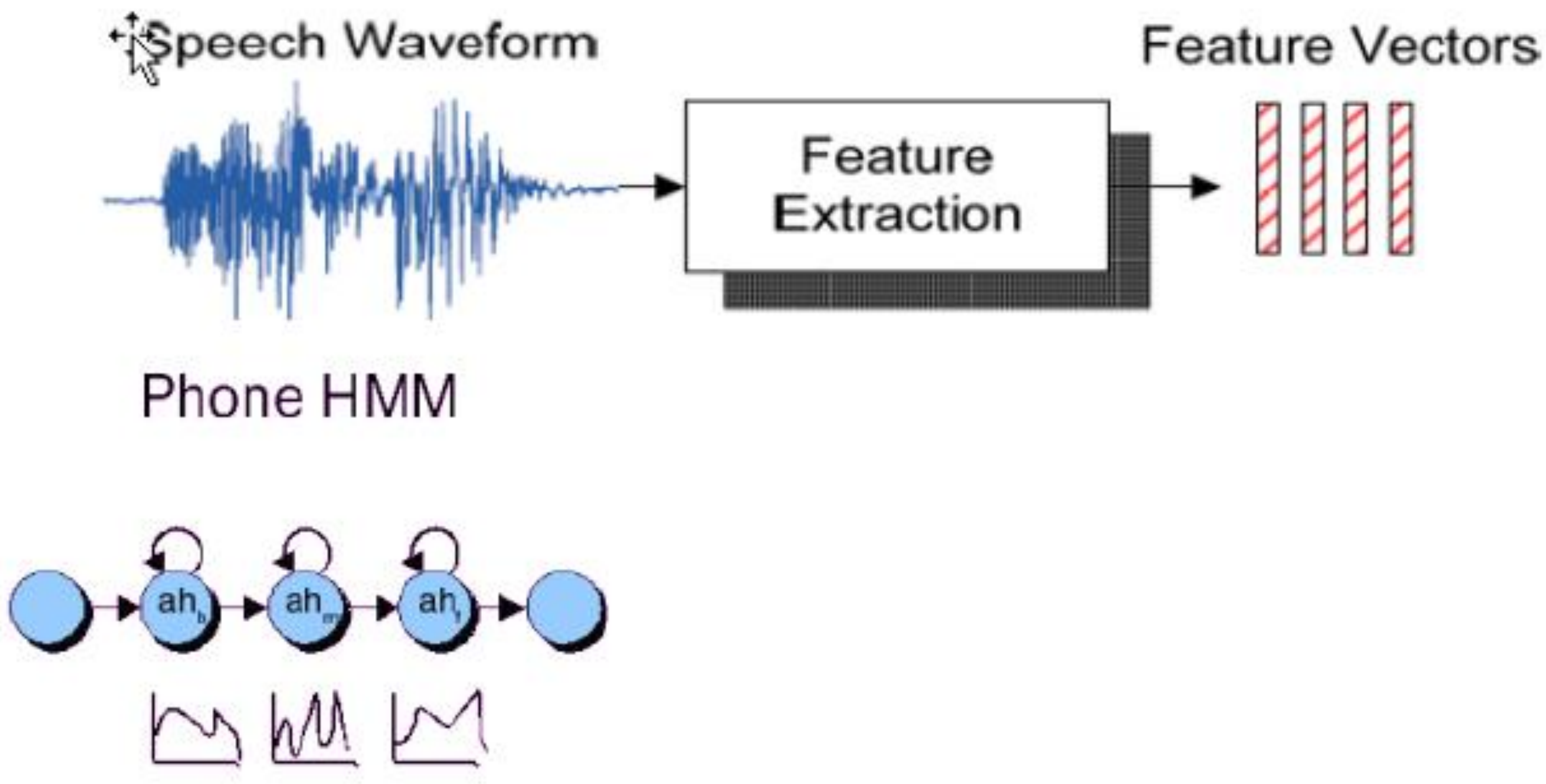
How the CNN "sees" an unpaved road. Top: subset of the camera image sent to the CNN. Bottom left: Activation of the first layer feature maps. Bottom right: Activation of the second layer feature maps. This demonstrates that the CNN learned to detect useful road features on its own, i. e., with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads.

CNN network



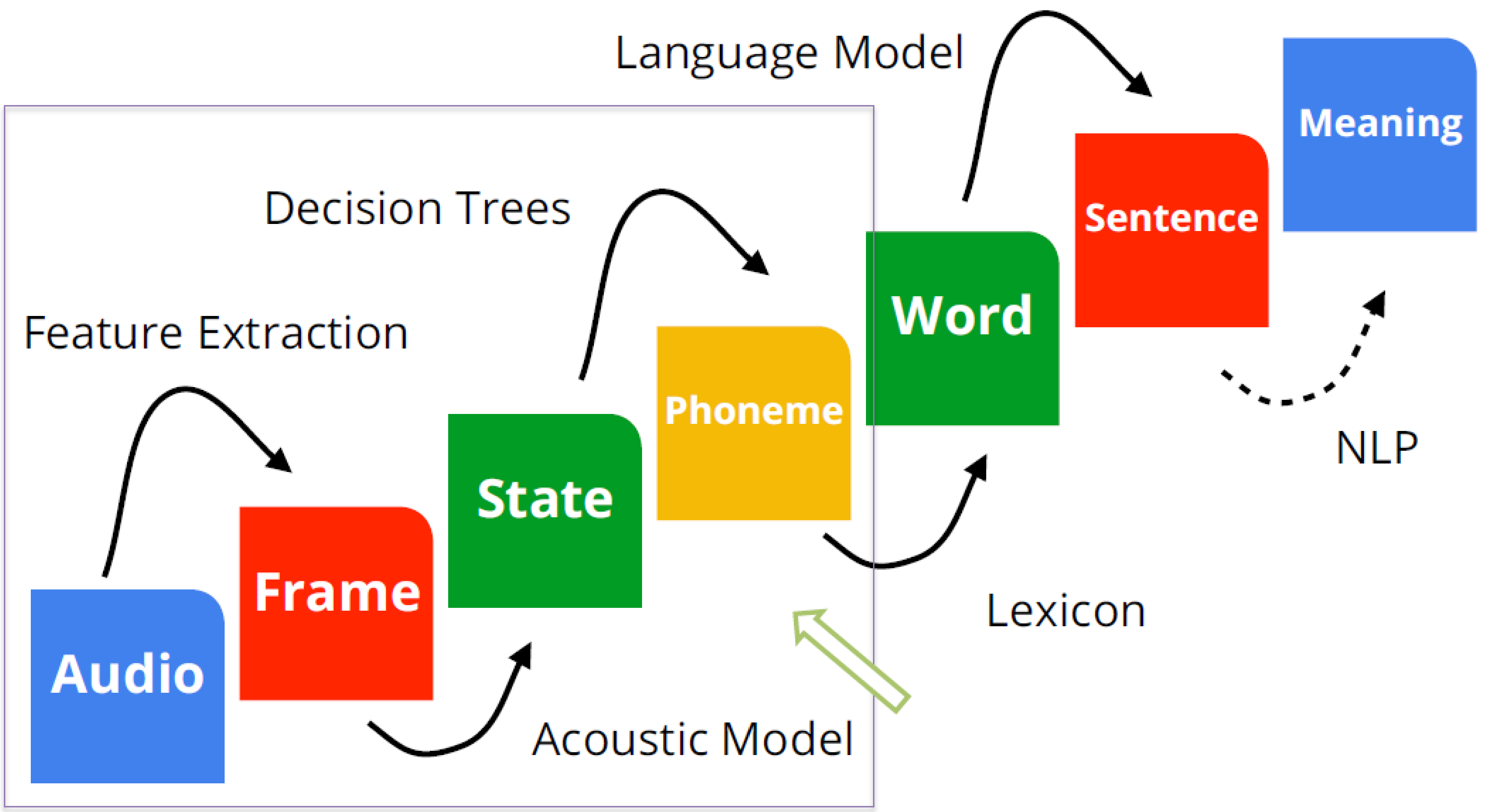
CNN architecture. The network has about 27 million connections and 250 thousand parameters.

Automated Speech Recognition



$$\widehat{W} = \mathrm{argmax}_w P(W|O)$$

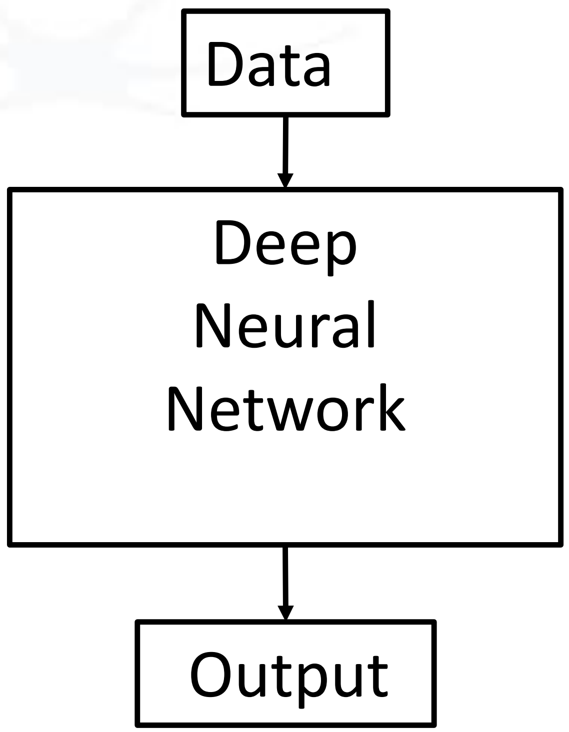
Automated Speech Recognition



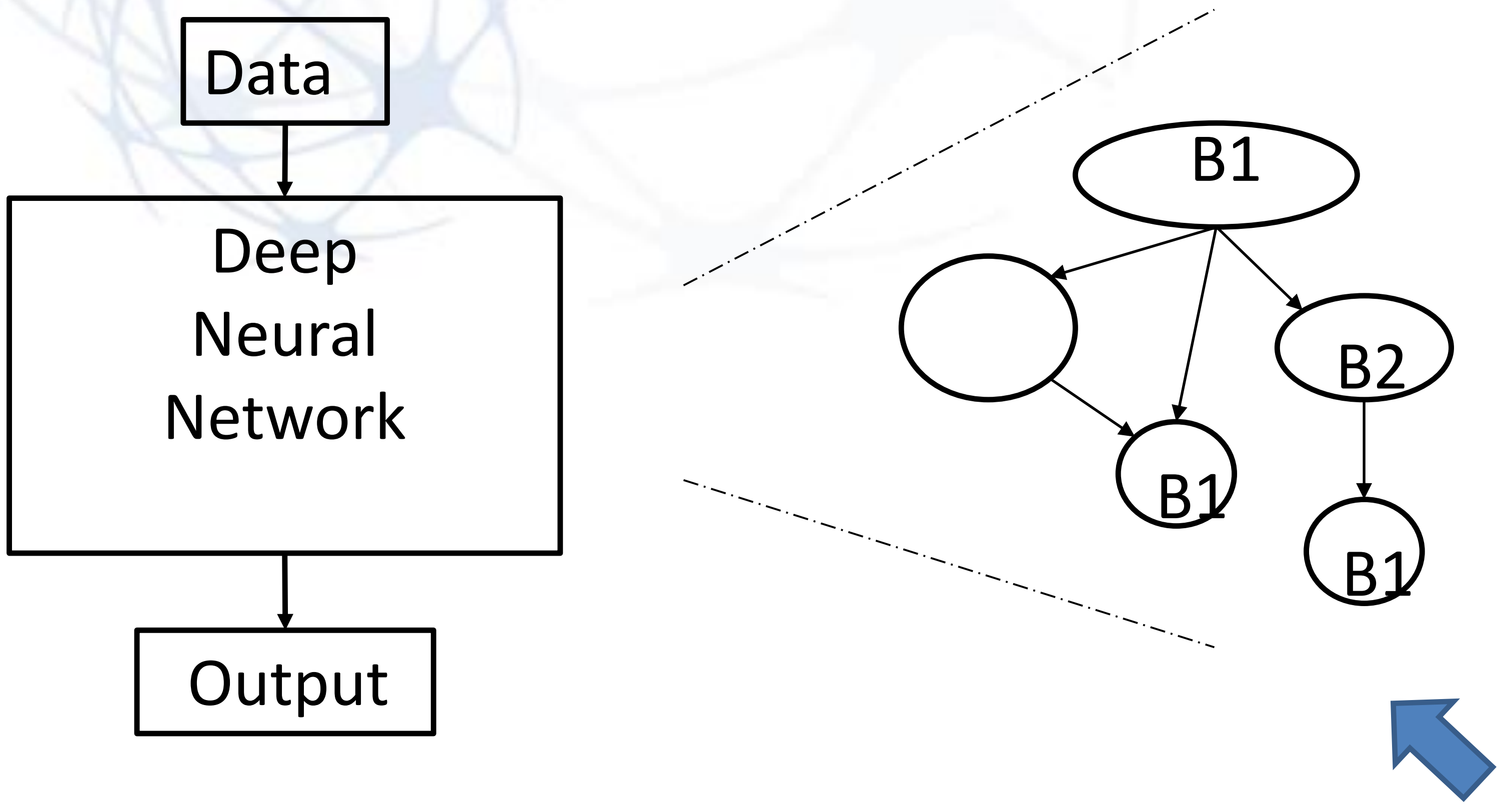
* Slide from V. Vanhoucke, ICML 2013 Keynote

End-To-End DNN

- Idealized Framework



DNN Inspection & Introspection



B1 -parallel

B2 –latency sensitive

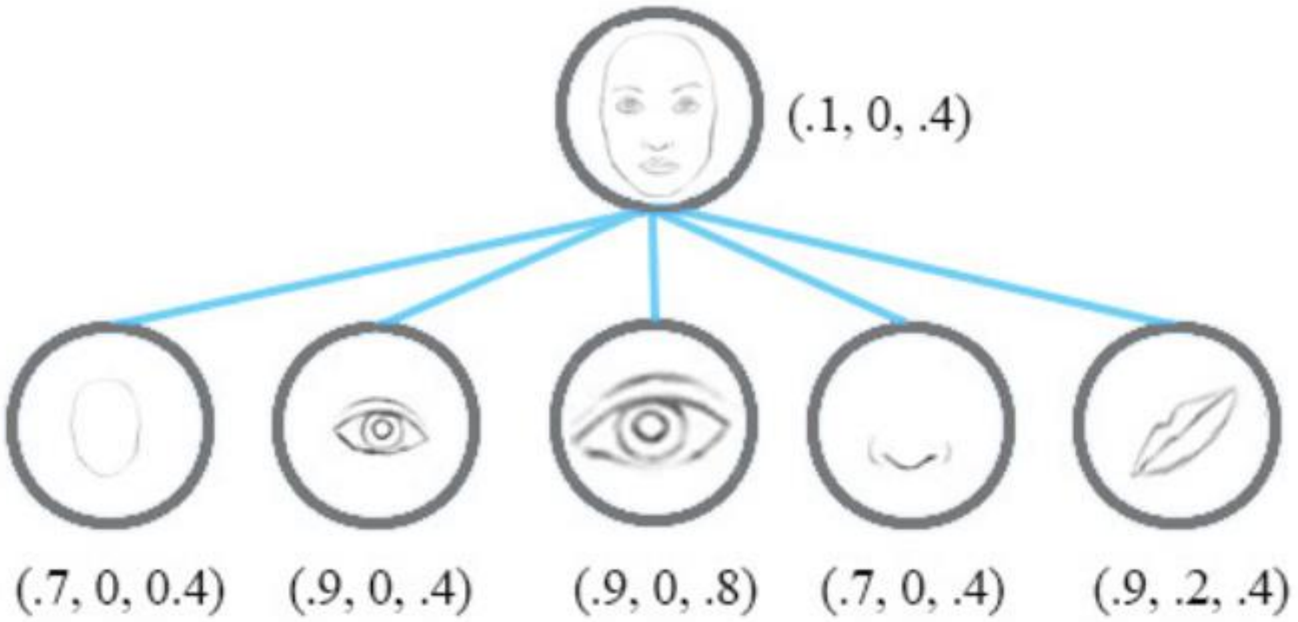
B3 – allows low precision

Annotated dataflow graph

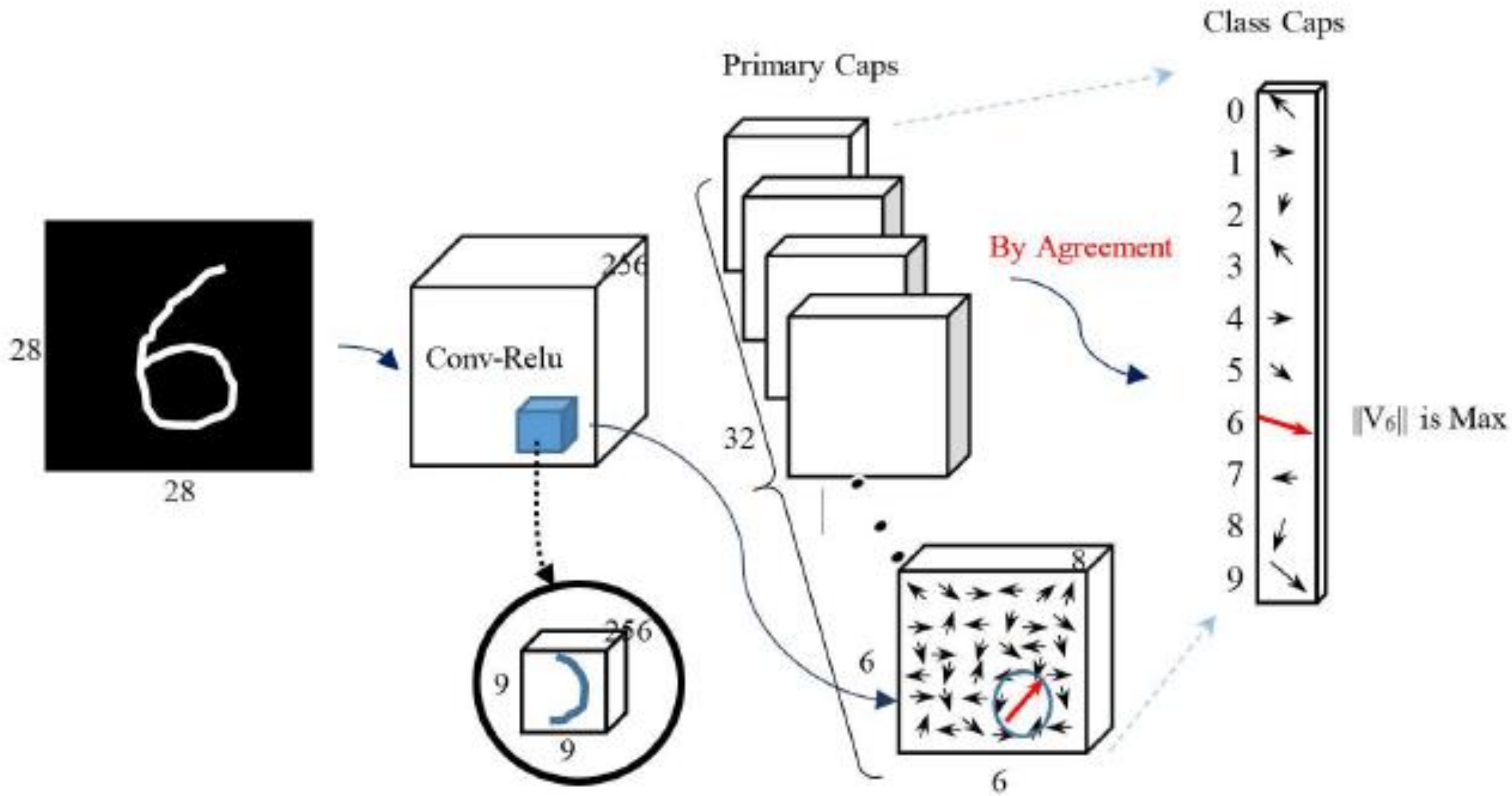
Capsule Networks

Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." *Advances in Neural Information Processing Systems*. 2017.

A capsule is a group of neurons that not only capture the likelihood but also the parameters of the specific feature.



Capsule Networks connectivity



Conjecture2

- Capsule Networks enable improved introspection
  - Tailored HW and SW stack
    - Precision –
      - how many precision bits? Inter-phase data communication
    - Interconnection – data flow
    - Code Generation
    - Reliability – tolerance to HW failure
    - Resilience – resist adversarial data

ML Application Components

- Data Preparation

    acquiring, producing, cleaning

    ENOUGH data to feed ML algorithm

- Feature Selection and Extraction

    identify data characteristics and behaviors

    of interest: what data aspects are important

- Productization/Deployment

    deploy a stable system at SCALE;

    deal with data variations over time

(model drift: phenomena evolve & models don't)

End-to-End ML Components

-The **data pipeline**

clean, perhaps labeled, accessible dataset;

message queue,

storage,

preprocessing (such as normalization and vectorization)

-The choice of **algorithms and their tuning**

choice of deep network topology

hyper-parameter optimization

-The **hardware** associated with the training of algorithms;

- Visualization / actuation/ communication **of results**

Data Preparation Challenges

- End-to-End DNN -> handling multi-modal data types
  - Video, Audio, Streaming, IoT, etc…
- Handling Data Artifacts
  - Data Normalization

Protocol Buffers as Canonical Data Representation

- Protocol Buffers
  - Standardized data transfer format for data centers

- DER (Distinguished Encoding Rules)

Research Approach

- Canonical Data Representation

- Inspect/Introspection of Trained Network

- Global Code Generation Approach
  - Partitioned/Optimized Dataflow Graph
  - Multi-pass graph rewrite
- Related Work: TVM, Dawn

Software Stack

Data Ingestion & Canonical Representation

Graph Optimizer – connectivity, operator merging

Tensor-Level Optimizer – memory, precision, schedule

JIT Runtime

ISA

Graph Optimizer

- Computation
  - Operator Merging
  - Precision
    - Bit-width determination
    - Data Transducer
- Memory
  - Data Access
  - Data Layout
  - Reuse
- Partitioning & Scheduling

IR

- Potential Reuse of

Halide, TVM, Weld, Tensorflow
XLA, Intel GraphN, DLVM, Glow, DLVM

Memory Optimization Considerations

- CPU – multi-level caches
- GPU – local(shared) memory, L2, L3

Protocol buffer access

Operators:

scalar

vectors

tensors

Tailoring Optimizations

- Bit-width adaptation
- Precision determination pass
- Redundancy / Resiliency pass
- Multi-precision accumulation pass

- Phase Ordering Problem

Conclusion

- Research topics:
  - Full-stack: HW, SW, Application, system
  - Cross-layer optimization
  - Societal Applications