Practical Deep Learning — Intel Course (WEEK — 05) — Hot Research

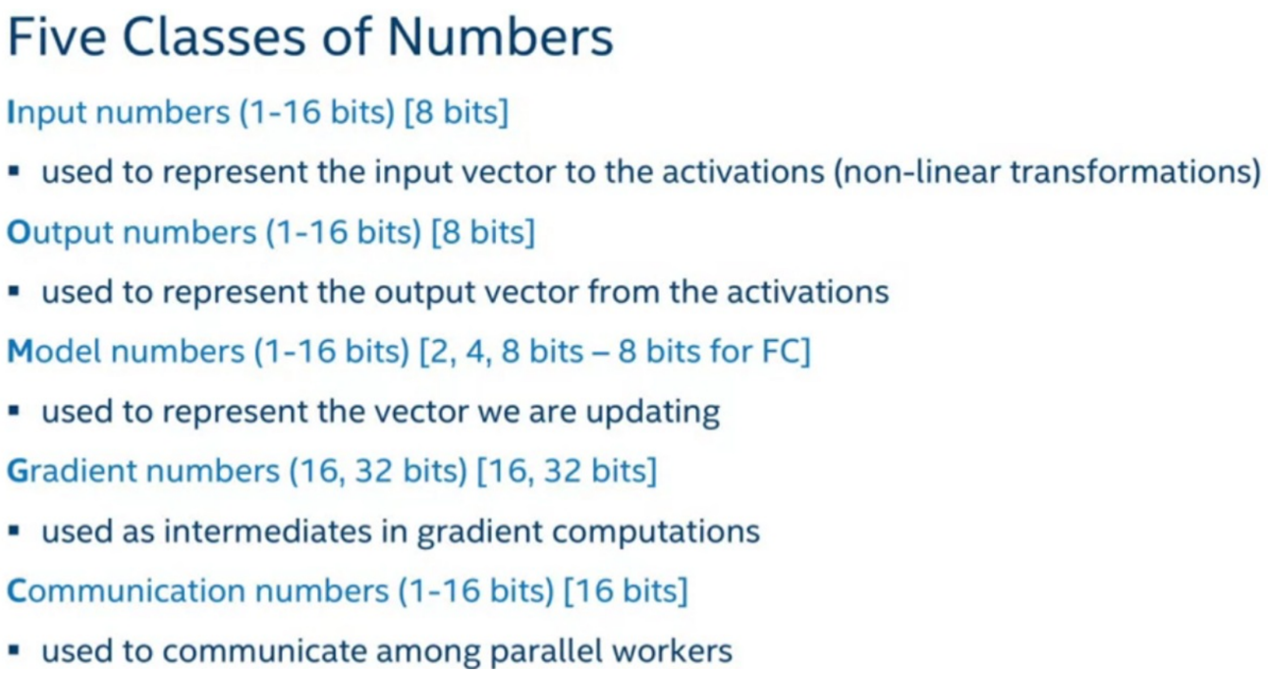
medium.com/@aakashgoel12/practical-deep-learning-intel-course-week-05-hot-research-9d771ae1ffc7

July 14, 2020

> Low bit Precision

Most DL practitioners use 32 bits of floating point precision to train and test deep networks. However, such high precision is not required especially in testing and deploying deep networks.

- Input numbers (vectors) — 8 bit precision
- Output numbers (vectors) — 8 bit precision
- Model numbers (weights) — 8 bit precision
- Gradient numbers — 16,32 bit precision (require most precision)
- Communication numbers (Distribute training across multiple nodes, gradients are communicated) — 16 bits



Five Classes of Numbers

Input numbers (1-16 bits) [8 bits]

- used to represent the input vector to the activations (non-linear transformations)

Output numbers (1-16 bits) [8 bits]

- used to represent the output vector from the activations

Model numbers (1-16 bits) [2, 4, 8 bits – 8 bits for FC]

- used to represent the vector we are updating

Gradient numbers (16, 32 bits) [16, 32 bits]

- used as intermediates in gradient computations

Communication numbers (1-16 bits) [16 bits]

- used to communicate among parallel workers

> GAN

Refer — nips 2016 tutorial GAN IAN Goodfellow (https://arxiv.org/abs/1701.00160)

Group of researchers added noise in image and whatever appears to naked eye isn't classified by CNN. Example — given image of bus (appearing via naked eye) classified as **Ostrich.**

Two types of Model:

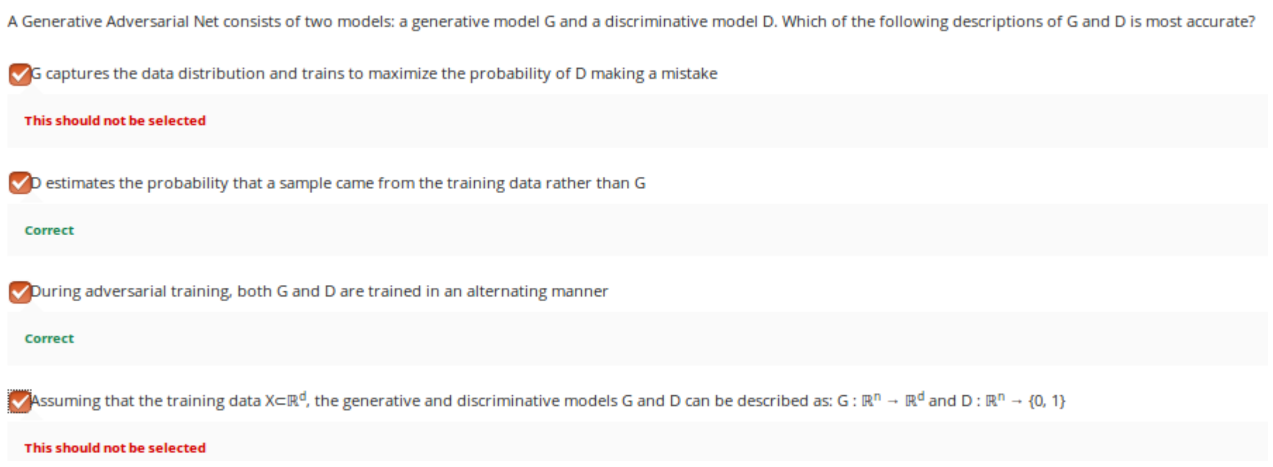
- CNN model discriminates b/w real and fake image
- CNN model generates image itslef

Dataset

- C14 — car, horses, plan,
- LSUN — Bed room

Application

- Text to image
- Single Image Super-Resolution
- Adobe created interactive GAN which allows user to create rough sketch of image such as black triangle and few green lines to produce image of mountain with grassy field


A Generative Adversarial Net consists of two models: a generative model G and a discriminative model D. Which of the following descriptions of G and D is most accurate?

[✓ G captures the data distribution and trains to maximize the probability of D making a mistake]

This should not be selected

[✓ D estimates the probability that a sample came from the training data rather than G]

Correct

[✓ During adversarial training, both G and D are trained in an alternating manner]

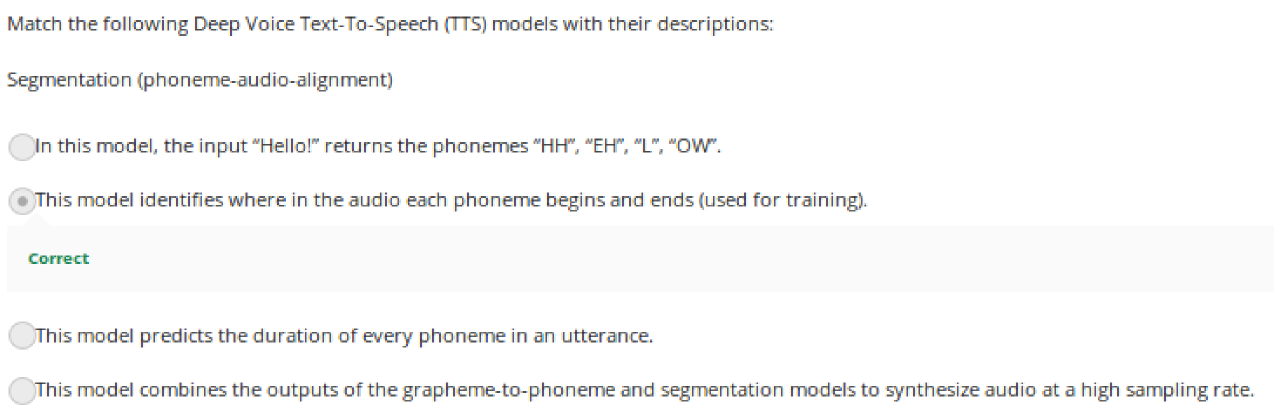
Correct

[✓ Assuming that the training data $X \subset \mathbb{R}^d$, the generative and discriminative models G and D can be described as: $G : \mathbb{R}^n \to \mathbb{R}^d$ and $D : \mathbb{R}^n \to \{0, 1\}$]

This should not be selected

> Deep Voice

Allows human computer interaction w/o requiring visual interfaces.

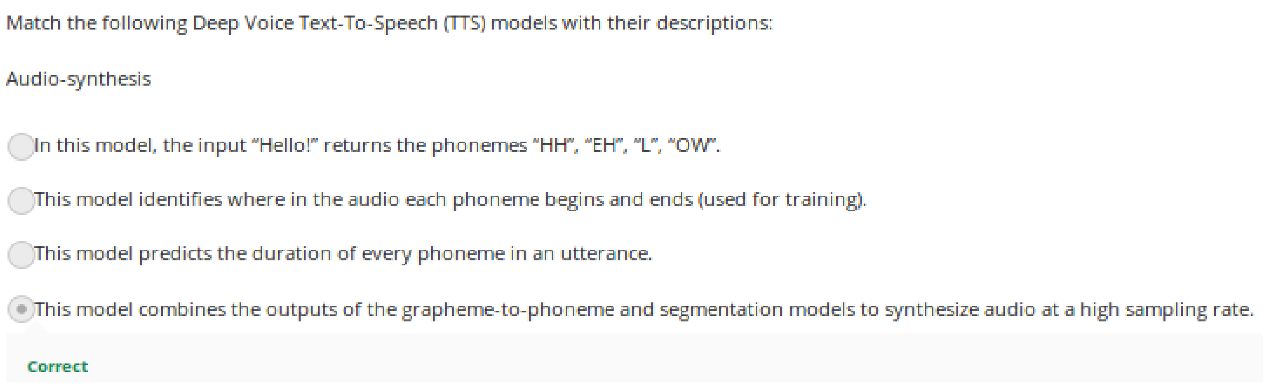

Match the following Deep Voice Text-To-Speech (TTS) models with their descriptions:

Segmentation (phoneme-audio-alignment)

- In this model, the input "Hello!" returns the phonemes "HH", "EH", "L", "OW".
- [✓ This model identifies where in the audio each phoneme begins and ends (used for training).]

Correct

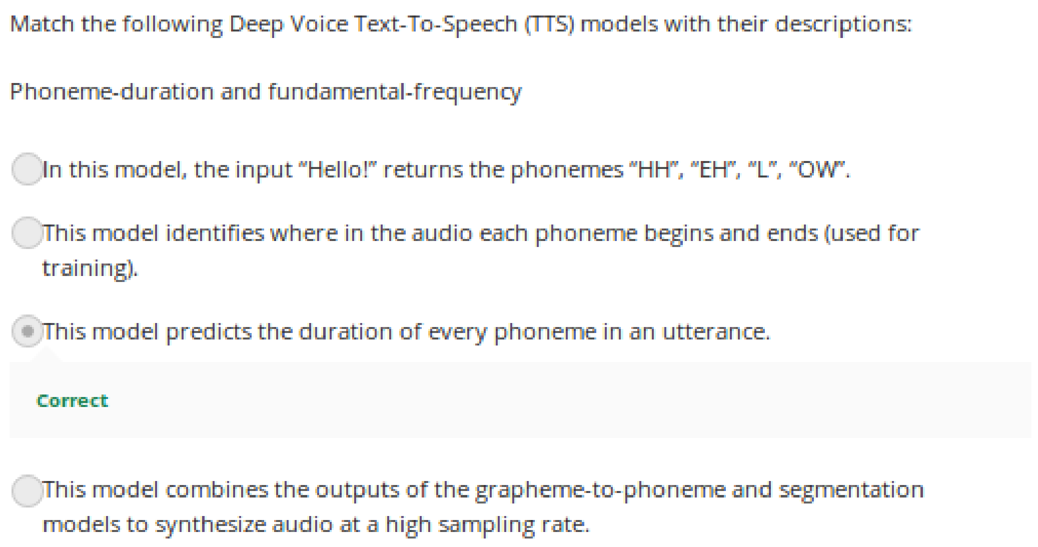
- This model predicts the duration of every phoneme in an utterance.
- This model combines the outputs of the grapheme-to-phoneme and segmentation models to synthesize audio at a high sampling rate.


Match the following Deep Voice Text-To-Speech (TTS) models with their descriptions:

Audio-synthesis

- In this model, the input “Hello!” returns the phonemes “HH”, “EH”, “L”, “OW”.
- This model identifies where in the audio each phoneme begins and ends (used for training).
- This model predicts the duration of every phoneme in an utterance.
- [✓ This model combines the outputs of the grapheme-to-phoneme and segmentation models to synthesize audio at a high sampling rate.]

Correct

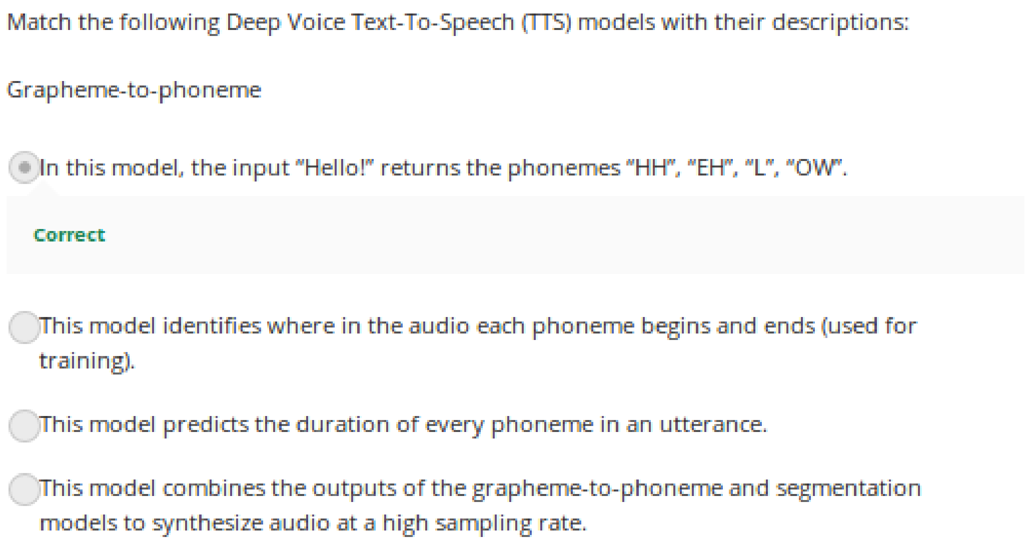

Match the following Deep Voice Text-To-Speech (TTS) models with their descriptions:

Phoneme-duration and fundamental-frequency

- In this model, the input “Hello!” returns the phonemes “HH”, “EH”, “L”, “OW”.
- This model identifies where in the audio each phoneme begins and ends (used for training).
- [✓ This model predicts the duration of every phoneme in an utterance.]

Correct

- This model combines the outputs of the grapheme-to-phoneme and segmentation models to synthesize audio at a high sampling rate.


Match the following Deep Voice Text-To-Speech (TTS) models with their descriptions:

Grapheme-to-phoneme

- [✓ In this model, the input “Hello!” returns the phonemes “HH”, “EH”, “L”, “OW”.]

Correct

- This model identifies where in the audio each phoneme begins and ends (used for training).
- This model predicts the duration of every phoneme in an utterance.
- This model combines the outputs of the grapheme-to-phoneme and segmentation models to synthesize audio at a high sampling rate.

> Reinforcement Learning

Bounce

> Additional resource

Speech Synthesis

WaveNet: A Generative Model for Raw Audio: https://arxiv.org/abs/1609.03499

This paper introduces WaveNet, a deep neural network for generating raw audio waveforms. The model is fully probabilistic and autoregressive, with the predictive distribution for each audio sample conditioned on all previous ones. When applied to text-to-speech, it yields state-of-the-art performance for both English and Mandarin.

Deep Voice 3: https://arxiv.org/abs/1710.07654

We present Deep Voice 3, a fully-convolutional attention-based neural text-to-speech (TTS) system. Deep Voice 3 matches state-of-the-art neural speech synthesis systems in naturalness while training ten times faster. We scale Deep Voice 3 to data set sizes unprecedented for TTS, training on more than eight hundred hours of audio from over two thousand speakers. We also describe how to scale inference to ten million queries per day on one single-GPU server.

Generative Adversarial Networks

NIPS 2016 Tutorial: Generative Adversarial Networks: https://arxiv.org/abs/1701.00160

The tutorial describes: (1) Why generative modeling is a topic worth studying, (2) how generative models work, and how GANs compare to other generative models, (3) the details of how GANs work, (4) research frontiers in GANs, and (5) state-of-the-art image models that combine GANs with other methods. Finally, the tutorial contains three exercises for readers to complete, and the solutions to these exercises.

Wasserstein GAN: https://arxiv.org/abs/1701.07875

We introduce a new algorithm named WGAN, an alternative to traditional GAN training. In this new model, we show that we can improve the stability of learning, get rid of problems like mode collapse, and provide meaningful learning curves useful for debugging and hyperparameter searches. Furthermore, we show that the corresponding optimization problem is sound, and provide extensive theoretical work highlighting the deep connections to other distances between distributions.

Reinforcement Learning

Human-level control through deep reinforcement learning: https://www.nature.com/articles/nature14236

We use recent advances in training deep neural networks to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. We tested this agent on the challenging domain of classic Atari 2600 games. We demonstrate that the deep Q-network agent, receiving only the pixels and the game score as inputs, was able to surpass the performance of all previous algorithms and achieve a level comparable to that of a professional human games tester across a set of 49 games, using the same algorithm, network architecture and hyperparameters.

Mastering the game of Go with deep neural networks and tree search: https://www.nature.com/articles/nature16961

We introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0.

Image Generator

A Neural Algorithm of Artistic Style: https://arxiv.org/abs/1508.06576

We introduce an artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images.

Pixel Recurrent Neural Networks: https://arxiv.org/abs/1601.06759

We present a deep neural network that sequentially predicts the pixels in an image along the two spatial dimensions. Our method models the discrete probability of the raw pixel values and encodes the complete set of dependencies in the image. Samples generated from the model appear crisp, varied and globally coherent.

Robotics

End-to-End Training of Deep Visuomotor Policies: https://arxiv.org/abs/1504.00702

We develop a method that can be used to learn policies that map raw image observations directly to torques at the robot's motors. The policies are represented by deep convolutional neural networks (CNNs) with 92,000 parameters, and are trained using a partially observed guided policy search method, which transforms policy search into supervised learning, with supervision provided by a simple trajectory-centric reinforcement learning method. We evaluate our method on a range of real-world manipulation tasks that require close coordination between vision and control, such as screwing a cap onto a bottle, and present simulated comparisons to a range of prior policy search methods.

Training

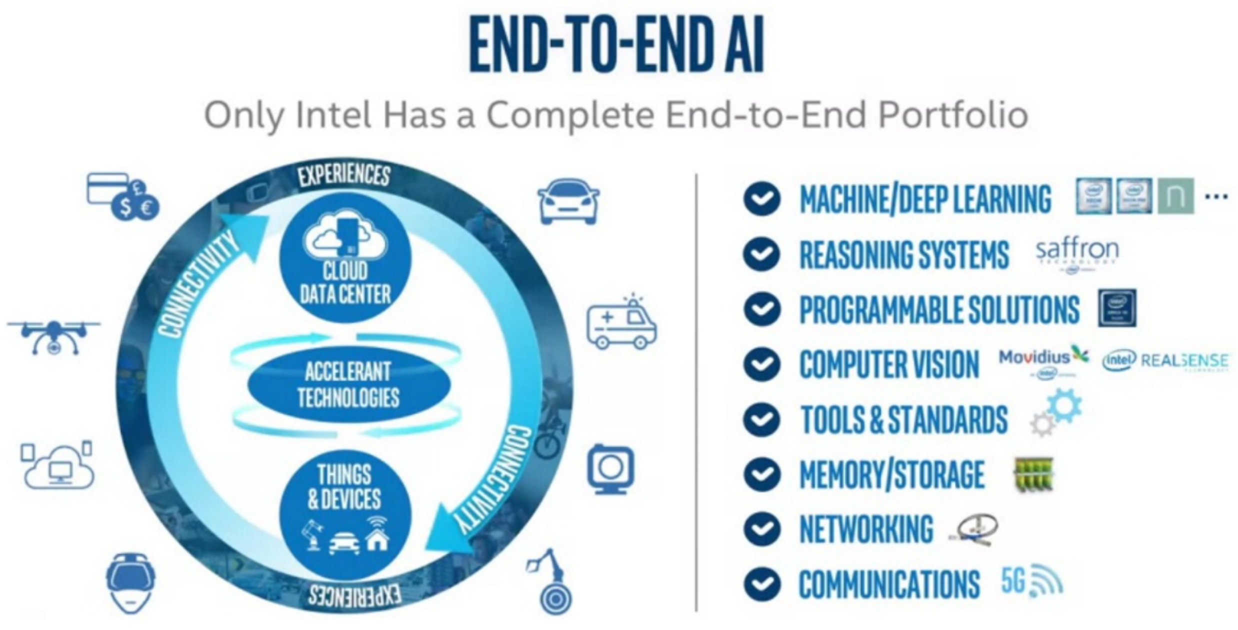
Distilling the knowledge in a neural network: https://arxiv.org/abs/1503.02531

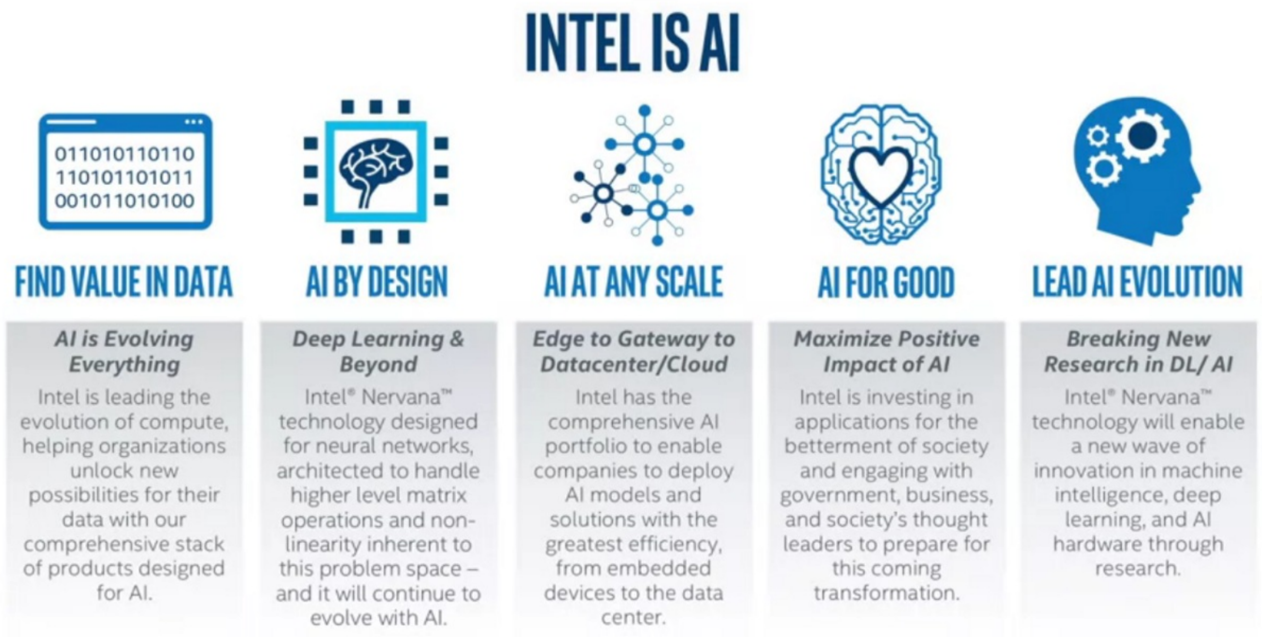
We achieve some surprising results on MNIST and we show that we can significantly improve the acoustic model of a heavily used commercial system by distilling the knowledge in an ensemble of models into a single model. We also introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

> *Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models*: https://arxiv.org/abs/1702.03275

We propose Batch Renormalization, a simple and effective extension to ensure that the training and inference models generate the same outputs that depend on individual examples rather than the entire minibatch. Models trained with Batch Renormalization perform substantially better than batchnorm when training with small or non-i.i.d. minibatches. At the same time, Batch Renormalization retains the benefits of batchnorm such as insensitivity to initialization and training efficiency.

> **Intel AI**





INTEL IS AI

FIND VALUE IN DATA

AI is Evolving Everything

Intel is leading the evolution of compute, helping organizations unlock new possibilities for their data with our comprehensive stack of products designed for AI.

AI BY DESIGN

Deep Learning & Beyond

Intel® Nervana™ technology designed for neural networks, architected to handle higher level matrix operations and non-linearity inherent to this problem space – and it will continue to evolve with AI.

AI AT ANY SCALE

Edge to Gateway to Datacenter/Cloud

Intel has the comprehensive AI portfolio to enable companies to deploy AI models and solutions with the greatest efficiency, from embedded devices to the data center.

AI FOR GOOD

Maximize Positive Impact of AI

Intel is investing in applications for the betterment of society and engaging with government, business, and society's thought leaders to prepare for this coming transformation.

LEAD AI EVOLUTION

Breaking New Research in DL/ AI

Intel® Nervana™ technology will enable a new wave of innovation in machine intelligence, deep learning, and AI hardware through research.

QUIZ

1. Imagine that you are looking at numerous images of animals at the zoo and are interested in training a model that correctly classifies each image by animal. You also want to generate a bounding box around the animal in the image.

   Which of the following categories of machine learning algorithms would this model fit under?

   ● Supervised

   ○ Unsupervised

   ○ Semi-supervised

   ○ Reinforcement

2. Imagine that you are looking at numerous images of animals at the zoo and are interested in training a model that correctly classifies each image by animal. You also want to generate a bounding box around the animal in the image.

   What of the following Deep Learning models could you train to classify each image by animal?

   - [ ] Recurrent Neural Network (RNN)
   - [x] Feed Forward Network
   - [x] Convolutional Neural Network (CNN)
   - [ ] Long Short Term Memory (LSTM)

3. Which of the following steps are used when training a neural network?

- [✓ Cost]
- [✓ Forward-pass]
- ☐ Update activation function
- [✓ Update weights]
- [✓ Randomly seed weights]
- [✓ Backward-pass]
- ☐ Add units/neurons

4. Which of the following steps are used when testing a neural network?

- [✓ Cost]
- [✓ Forward-pass]
- ☐ Update activation function
- ☐ Update weights
- ☐ Randomly seed weights
- ☐ Backward-pass
- ☐ Add units/neurons

5. Which of the following steps are used when running inference with a neural network (using a model on new data)?

- [ ] Cost
- [x] Forward-pass
- [ ] Update activation function
- [ ] Update weights
- [ ] Randomly seed weights
- [ ] Backward-pass
- [ ] Add units/neurons

6. Which of the following layers are found in Convolutional Neural Networks (CNN)?

- [ ] LSTM (Long Short Term Memory)
- [ ] GRU (Gated Recurrent Unit)
- [x] Max-Pooling
- [x] Affine

7. Which of the following layers are found in Feed Forward Networks?

- [ ] LSTM (Long Short Term Memory)
- [ ] GRU (Gated Recurrent Unti)
- [ ] Max-Pooling
- [x] Affine

8. If there are *n* inputs to a 1-D Convolutional Neural Network, and there are two filters, the filter size is 2, the stride is 2 and the padding is 1, then how many units are in the first hidden layer? Assume *n* is even.

○ n

[✓ $n + 2$]

○ $2n$

○ $2n + 2$

9. When fine-tuning a Convolutional Neural Network, which of the following techniques is most effective?

- ○ Start by training a network with a very large dataset, and then modify and re-learn the first layer(s) with a smaller dataset after the network learns a mapping from pixels to a feature space
- [✓ Start by training a network with a very large dataset, and then modify and re-learn the last layer(s) with a smaller dataset after the network learns a mapping from pixels to a feature space]
- ○ Start by training a network with a small dataset, and then modify and re-learn the first layer(s) with a very large dataset after the network learns a mapping from pixels to a feature space
- ○ Start by training a network with a small dataset, and then modify and re-learn the last layer(s) with a very large dataset after the network learns a mapping from pixels to a feature space

10. Imagine that you are trying to create captions for a given input image and want to use Deep Learning.

Which of the following models should you use?

- ○ One deep vision and language generating CNN
- ○ A deep vision RNN fed into a language generating CNN
- [✓ A deep vision CNN fed into a language generating RNN]
- ○ A language generating RNN fed into a deep vision CNN

11. Imagine that you are trying to create captions for a given input image and want to use Deep Learning.

You do not have very many image-caption pairs, but you do remember a way to train the CNN portion without needing a large amount of domain-specific data. Which of the following techniques should you use?

- ○ Gradient descent with a high learning rate (to make up for the lack of data)
- [✓ Transfer learning via fine-tuning, as the last layers of a model are task specific]
- ○ Forward-propagation of generic images (to train the model on general image recognition/classification tasks)
- ○ The LSTM network architecture (to detect temporal dependencies that can compensate for small amounts of data)

12. When encountering exploding gradients during training, which of the following methods effectively counteracts the exploding gradients?

- ☐ Use LSTM and GRU networks
- ☑ Use RMSprop to adjust the learning rate
- ☑ Clip the gradients at some threshold
- ☑ Truncate BPTT

13. What are some possible issues with asynchronous stochastic gradient descent?

- [✓ Asynchronous stochastic gradient descent requires more epochs to train]
- Asynchronous stochastic gradient descent does not overcome communication delays
- [✓ Asynchronous stochastic gradient descent requires more hypertuning of hyperparameters, including momentum and learning rate]
- [✓ Asynchronous stochastic gradient has not been shown to scale and retain accuracy on large models]

14. Which of the following characteristics are true of a data-parallel approach?

- [✓ The algorithm distributes the data between various cores]
- Each core is responsible for estimating different parameter(s)
- [✓ A data-parallel approach is useful when there are smaller number of nodes in the cluster]
- A data-parallel approach is useful when the number of parameters to be estimated is large

15. Which of the following activation functions is most commonly used in Deep Learning models today (in terms of the frequency of usage), excluding the last layer of a network?

- Logistic, Sigmoid
- Softmax
- Tanh
- [✓ Rectified Linear Unit]

17. In the very first exercise, we used a Feed Forward Network as shown below:

```
from neon.layers import Affine
from neon.transforms import Rectlin, Softmax

layers = []
layers.append(Affine(nout=10, init=init_norm, activation=Rectlin()))
layers.append(Affine(nout=10, init=init_norm,
                     activation=Softmax()))
```

As a reminder, this model took in as input MNIST images, which are 28 by 28 pixels in size. The goal was to predict which of 10 digits each image corresponded to.

How many inputs does this network have?

784

18. In the very first exercise, we used a Feed Forward Network as shown below:

```
from neon.layers import Affine
from neon.transforms import Rectlin, Softmax

layers = []
layers.append(Affine(nout=10, init=init_norm, activation=Rectlin()))
layers.append(Affine(nout=10, init=init_norm,
                     activation=Softmax()))
```

As a reminder, this model took in as input MNIST images, which are 28 by 28 pixels in size. The goal was to predict which of 10 digits each image corresponded to.

How many parameters does this network have (including weights and biases)?

7960

19. In the very first exercise, we used a Feed Forward Network as shown below:

```
from neon.layers import Affine
from neon.transforms import Rectlin, Softmax

layers = []
layers.append(Affine(nout=10, init=init_norm, activation=Rectlin()))
layers.append(Affine(nout=10, init=init_norm,
                     activation=Softmax()))
```

As a reminder, this model took in as input MNIST images, which are 28 by 28 pixels in size. The goal was to predict which of 10 digits each image corresponded to.

If we change the model so it takes as input images of size 56 by 56 pixels (and leave the hidden layer and output as is), how many more parameters does the network have?

31480

19 is incorrect answer