

VIRTUALIZING MACHINE
LEARNING WORKLOADS WITH
NVIDIA GPUs PROVIDES THE BEST
OF BOTH WORLDS

Contents

Introduction	3
What is Machine Learning?	3
Why VMware and Machine Learning?	4
VMware is Making Strides to Improve GPU Performance.....	4
General Purpose GPU (GP GPU)	5
TensorFlow	6
TensorFlow Setup.....	6
Real-World Machine Learning Use Cases	7
Use Case #1: Image Processing	7
Use Case #2: Stock Market Prediction	9
Use Case #3: Closed Captioning	13
Conclusion	14
References	15
Acknowledgements	15
Authors	15
Contributors.....	15
Appendix A: TensorFlow Installation in Centos 7 Virtual Machine.....	16
Appendix B: Python Code for Style Transformation Use Case	16
Appendix C: Python Code for Stock Prediction Use Case	18
Appendix D: Python Code for Image Captioning Use Case.....	22

Introduction

VMware® engaged VLSS®, a VMware solution partner, to demonstrate how virtualizing Machine Learning (ML) applications using VMware significantly speeds up performance. As part of this engagement, students at the University of California, Berkeley participated in the project by developing three real-world Machine Learning use cases.

This project brief demonstrates how the use of new technologies can offer more benefits than today's technologies offered by Amazon®, Google®, and Microsoft®.

Machine Learning is a hot topic and every size company must leverage its power to remain competitive. Over the past few years, VMware has made substantial investments in improving vSphere® performance to support Machine Learning. This engagement clearly demonstrates that VMware can run and execute a machine learning model in a matter of minutes compared to other popular platforms that can take days or weeks.

What is Machine Learning?

"Machine learning is a subset of Artificial Intelligence (AI) in the field of computer science that often uses statistical techniques to give computers the ability to 'learn' with data, without being explicitly programmed."¹

In an environment where everyone is concerned with personal and data security, ML has multiple uses in the areas of face detection, face recognition, image classification, and speech recognition. Software security companies use ML in the fight to stop computer viruses/spam and to detect fraud. Organizations such as Google and its competitors use it for on-line search. There are even more applications in financial services, healthcare, and customer service. For example, Amazon® uses ML to improve customer experiences in key areas of their business, including product recommendations, substitute product prediction, fraud detection, meta-data validation, and knowledge acquisition."² From predicting the weather to diagnosing diseases to driving cars, ML has many applications.

According to IDC, spending on AI and ML is to expected to grow from \$12B in 2017 to \$57.6B by 2021. According to a MarketsAndMarkets report, the market size for machine learning is expected to grow from \$1.03 Billion in 2016 to **\$8.81 Billion** by 2022, at a Compound Annual Growth Rate (CAGR) of 44.1 percent during the forecast period. Market growth will be fueled by continuous technological advancements and the exploding growth of data.

¹ https://en.wikipedia.org/wiki/Machine_learning

² <https://www.forbes.com/sites/louiscolumbus/2018/02/18/roundup-of-machine-learning-forecasts-and-market-estimates-2018/#49b162482225>

Why VMware and Machine Learning?

ML takes a lot of computational power, so the costs of servers to support ML can be expensive. To get around this problem, many organizations have moved their ML to the cloud with solutions offered by Amazon, Google, and Microsoft. With a cloud platform, you can spin up 10,000 machines and run them for an hour to get your answer, which is significantly less expensive than procuring thousands of servers.

Unfortunately, there are a few downsides to these ML cloud solutions. First, these solutions come with management tools that are cumbersome to use and take time to learn. Second, these solutions limit the applications to the one cloud environment and, if you have a hybrid- or multi-cloud deployment, you will have problems with performance, cost, and usability if your database, for example, is on-premise and your ML solution is in one of these public clouds.³

On the other hand, VMware offers a best-in-class hypervisor from a performance perspective and a popular, best-in-class management structure. Because VMware is a popular IT tool, your IT team is familiar with it. They don't need to learn anything new to leverage VMware for ML.

VMware is Making Strides to Improve GPU Performance

There has been a rapid evolution of technology to support AI and ML with the use of hardware accelerators. To that end, VMware has been working with accelerator vendors, such as NVIDIA®, Intel®, and AMD®, to create a shared, high performance platform for ML. VMware is also making strides to improve the performance of the GPU (Graphical Processor Unit) and virtualize that to one or more machines, something that is impossible to do in the Amazon and Google environment.

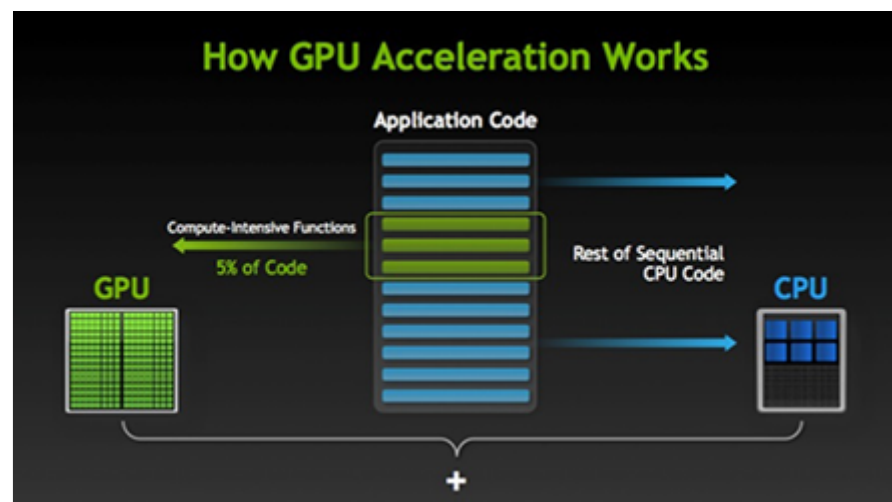


Figure 1. How GPU Acceleration Works (Source: NVIDIA)

³ <https://techbeacon.com/machine-learning-cloud-how-it-can-help-you-right-now>

Deep learning and the enhancement of Hardware Compute Accelerators (HCAs), such as GPUs, has fast-tracked the adoption of ML applications across a broad spectrum of real-world use cases. These include applications such as image processing, cancer diagnosis, stock prediction, self-driving technology, and text and speech recognition.

Accelerators are designed to handle massive floating-point computations that can run in parallel on many processing cores. The use of accelerators is a strong trend in HPC virtualization, particularly for accelerating deep learning workloads. They are most often used in conjunction with the CPUs of the node to accelerate certain compute-intensive functions that require a large amount of algebraic operations. Accelerators can significantly improve performance if your application is spending a lot of time doing a task that can be easily divided into parallel tasks or if many algebraic operations are needed to process your data.

Typical hardware compute accelerators include GPGPU, Intel Xeon Phi®, field-programmable gate array (FPGA), NVIDIA, AMD MxGPU, Intel, and Xilinx®.

In general, compute accelerators can be configured in DirectPath I/O (passthrough) mode on VMware vSphere®, which allows a guest Operating System (OS) to directly access the device, essentially bypassing the hypervisor. Because of the shortened access path, the performance of applications accessing accelerators is as fast as if you were using a bare-metal system.

General Purpose GPU (GP GPU)

A general-purpose GPU (GPGPU) is a graphics processing unit that performs computer graphics operations traditionally handled by high-powered CPUs. GPUs are adept at performing parallel processing and can leverage thousands of cores for computation. GPUs operate at lower frequencies but have many times the number of cores than CPU sockets.

THE VIRTUALIZED GPU INFRASTRUCTURE SETUP

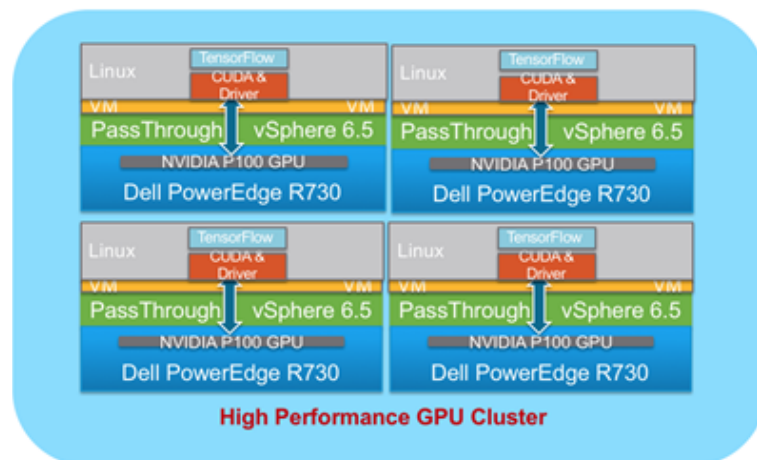


Figure 2. Logical Schematic of Virtualized GPU Infrastructure

The ML solution used a four-node vSphere cluster with Dell® R730 servers containing one NVIDIA Tesla® P100 card each. The GPUs were setup in pass-through mode for direct access from a TensorFlow™ VM. Each node had one TensorFlow Virtual Machine (VM) with dedicated access to the GPU card.

TensorFlow

Created by the Google Brain team, TensorFlow is a popular open source software library for high performance numerical computation. It comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many scientific domains. TensorFlow works across a multitude of platforms — from desktops to clusters of servers to mobile and edge devices — and can leverage hardware accelerators for its high-performance needs.

TensorFlow uses data flow graphs to perform numerical computations. It is highly flexible, portable, can handle complex computational frameworks, and supports multiple programming languages.⁴ The team used TensorFlow for developing and deploying real-world use cases for machine learning.

TensorFlow Setup

The team used Centos® 7.x as the Linux® OS for the TensorFlow VMs and followed the instructions at Linux TensorFlow Install to install TensorFlow components and the GPU version of TensorFlow, and the instructions from NVIDIA to download and install CUDA and cudnn. Lastly, the team templated the VM to use for cloning and deployment.

⁴ <https://opensourceforu.com/2017/01/best-open-source-machine-learning-frameworks/>

Real-World Machine Learning Use Cases

Three real-world machine learning use cases were chosen for this study, with the algorithms developed by the UC Berkeley students. The first two use cases leveraged a single instance TensorFlow for GPU-based processing, while the third use case was deployed on a distributed TensorFlow infrastructure.

Use Case #1: Image Processing

The Objective

The objective of this use case was to demonstrate that VMWare high-performance computing servers can speed up the time it takes to train and test a solution that emulates the artistic style of famous artists and “repaints” a photograph or picture with that style. Over time, the difference is slowly reduced and an image is discernible.

The Data

The algorithm was built off a pre-existing image-recognition model and only needed one content image and one style image to iteratively train.

The Algorithm

The architecture computed the difference between the generated image and image given for content, as well as the difference between the image containing the style and the generated image. For each iteration, it calculates both these differences and then learns iteratively how to minimize this difference.

The algorithm trained the network on 50 epochs with 10,000 individual images per epoch to maximize the output of the network.

The Results

The speed of the GPU-enabled virtual machines was exceptional and far outpaced conventional non-GPU virtual machines. Where a non-GPU-enabled VM took 500 hours to process 10,000 images, the GPU-enabled VM took 11 hours. The extreme computational capabilities of GPUs can be combined with the efficiency and sharing of vSphere environments to provide a robust infrastructure for machine learning workloads.

Style Transformation Platform and Examples

The Golden Gate Bridge image represents the content image (a photograph) and the style image is the the Starry Night painting by Vincent van Gogh. This machine learning training exercise took these two images with different ranges of weights and generated respective output images. The range of weights for content and style are different. The content weights go from 0 to 2 and the style weights go from 0 to 10.



Figure 3. Style transformation platform



Figure 4. Style transformation example 1

In Figure 4, we see a middle of the range style and content weight reflected in the image.

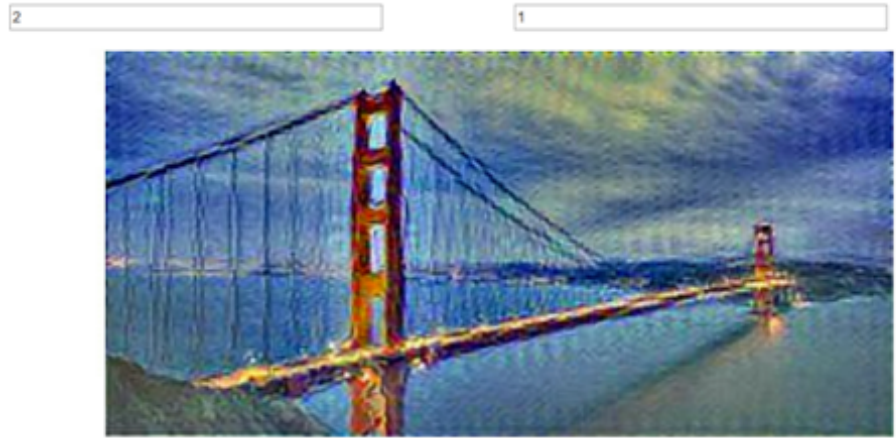


Figure 5. Style transformation example 2

In Figure 5, we see that the style weight is low and that the transformed image reflects most of the content image with less style.



Figure 6. Style transformation example 3

In Figure 6, we see that the style and content weight at their maximum, and this is reflected in the transformation.

Use Case #2: Stock Market Prediction

The Objective

The objective of this use case was to predict the values of the S&P 500 stock market on August 31, 2017. The algorithm compiled historical data for five months and used the data for the machine learning process to tune the algorithm and predict the values of the stock on August 31st.

The Data

The data was collected via API from historical archives available in Google Finance. The dataset contains approximately 42,000 minutes of data ranging from April 2017 to August 2017, on 500 stocks from the total S&P 500 index price.

The Algorithm

The code creates variables that represent the network's input, the network's output, weights, and biases that were initialized prior to model training. There are three major building blocks for a typical Neural Network model that includes the input layer, the hidden layers, and the output layer.

The model consists of four Neural Network layers, with the first having 1024 neurons, the second having 512 neurons, followed by 256 and 128 neurons. The type of Neural Network used is a feedforward network. Using activation functions, the hidden layers were transformed. The next layer was the output layer, which was used to transpose the data. Lastly, the model used a cost function to measure the deviation between the networks. The optimizer takes care of computations pertaining to weights and biases during training. After setting up the code, the algorithm used this model to continue training the network and output predicted stock prices.

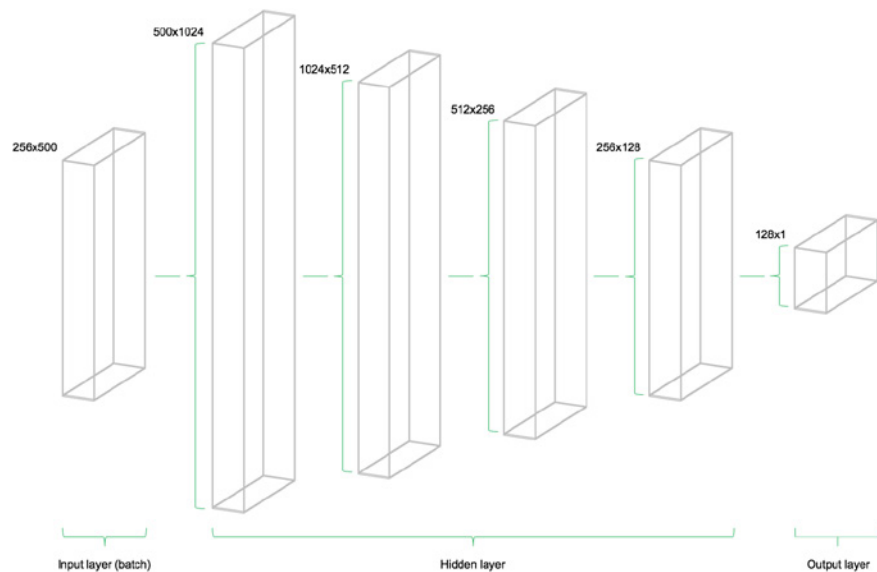
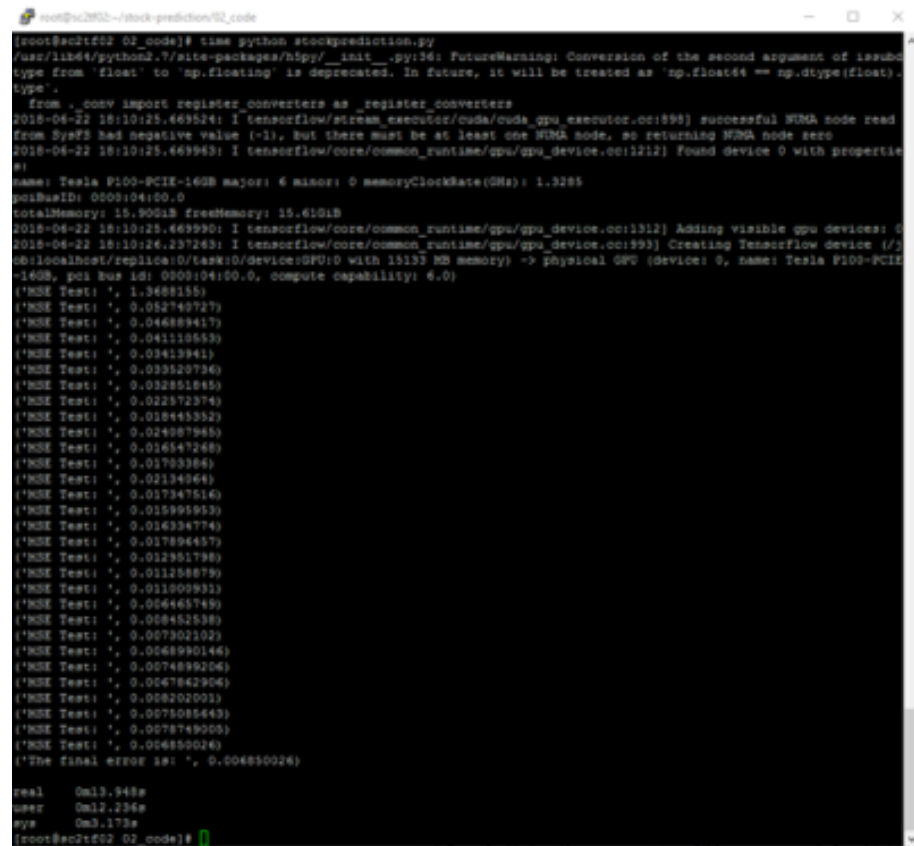


Figure 7. Feed Forward Neural Network layers used in stock price prediction⁵

⁵ Source: <https://medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877>

The Results

At the end of all iterations, the model outputs a final loss, with loss being defined as the mean squared error between the output of our model and expected output. The model is 99 percent accurate with less than a 1 percent error rate, as seen in Figures 8, 9, and 10 below.



```

[ec2@ec2-502 02_code]$ time python stockprediction.py
/usr/lib64/python3.7/site-packages/hlpy/_init_.py:14: FutureWarning: Conversion of the second argument of issubclass
type from 'float' to 'np.float64' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).
type'.
  from . conv import register_converters as _register_converters
2018-04-22 18:10:25.469524: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:1194] successful NvML node read
from SysFS had negative value (-1), but there must be at least one NvML node, so returning NvML node zero
2018-04-22 18:10:25.469943: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1121] Found device 0 with properties
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:04:00:0
totalMemory: 15.90GiB freeMemory: 15.61GiB
2018-04-22 18:10:25.469990: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1312] Adding visible gpu devices: 0
2018-04-22 18:10:26.237263: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1331] Creating TensorFlow device (/)
on local host / replica:0 / task:0 / device:GPU:0 with 15133 MB memory) -> [Physical GPU (Device: 0, name: Tesla P100-PCIE
-16GB, pci Bus Id: 0000:04:00:0, compute capability: 6.0)
('MSE Test: ', 1.3688155)
('MSE Test: ', 0.052740727)
('MSE Test: ', 0.046889417)
('MSE Test: ', 0.041110553)
('MSE Test: ', 0.03413941)
('MSE Test: ', 0.033520794)
('MSE Test: ', 0.032851845)
('MSE Test: ', 0.022572374)
('MSE Test: ', 0.018443352)
('MSE Test: ', 0.024087945)
('MSE Test: ', 0.016547248)
('MSE Test: ', 0.01703384)
('MSE Test: ', 0.02134044)
('MSE Test: ', 0.017947514)
('MSE Test: ', 0.015995953)
('MSE Test: ', 0.014334774)
('MSE Test: ', 0.017894457)
('MSE Test: ', 0.012301798)
('MSE Test: ', 0.011258879)
('MSE Test: ', 0.011000931)
('MSE Test: ', 0.054445749)
('MSE Test: ', 0.058452538)
('MSE Test: ', 0.057302102)
('MSE Test: ', 0.0548990144)
('MSE Test: ', 0.0574899204)
('MSE Test: ', 0.0567842304)
('MSE Test: ', 0.058202001)
('MSE Test: ', 0.0575085443)
('MSE Test: ', 0.0578749005)
('MSE Test: ', 0.054850026)
('The final error is: ', 0.004850026)

real    0m13.948s
user    0m12.234s
sys     0m3.173s
[ec2@ec2-502 02_code]$
  
```

Figure 8. Timed run with GPU

```

root@ac2tf01:~/stock_project/02_code
('MSE Test: ', 0.0027552482)
('The final error is: ', 0.0027552482)

real    0m14.235s
user    2m9.505s
sys     0m11.554s

[root@ac2tf01 02_code]# time python stockprediction.py
/usr/lib64/python2.7/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of issubclass from 'float' to 'np.floating' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from .conv import register_converters as _register_converters
2018-06-22 18:10:27.593125: E tensorflow/stream_executor/cuda/cuda_driver.cc:406] failed call to cuInit: CUDA_ERROR_NO_DEVICE: No device found
2018-06-22 18:10:27.593166: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:145] kernel driver does not appear to be running on this host (ac2tf01): /proc/driver/nvidia/version does not exist
('MSE Test: ', 2.3248402)
('MSE Test: ', 0.048509924)
('MSE Test: ', 0.032583077)
('MSE Test: ', 0.029590555)
('MSE Test: ', 0.028397785)
('MSE Test: ', 0.023785153)
('MSE Test: ', 0.025449033)
('MSE Test: ', 0.024014044)
('MSE Test: ', 0.02419918)
('MSE Test: ', 0.022346785)
('MSE Test: ', 0.021421447)
('MSE Test: ', 0.020637844)
('MSE Test: ', 0.021729138)
('MSE Test: ', 0.019519368)
('MSE Test: ', 0.020053254)
('MSE Test: ', 0.017756894)
('MSE Test: ', 0.019137895)
('MSE Test: ', 0.014735092)
('MSE Test: ', 0.01882456)
('MSE Test: ', 0.01424835)
('MSE Test: ', 0.015646045)
('MSE Test: ', 0.01484453)
('MSE Test: ', 0.02003589)
('MSE Test: ', 0.014252794)
('MSE Test: ', 0.013413045)
('MSE Test: ', 0.012794038)
('MSE Test: ', 0.012148801)
('MSE Test: ', 0.012375727)
('MSE Test: ', 0.01258977)
('MSE Test: ', 0.012676172)
('The final error is: ', 0.012676172)

real    0m14.268s
user    2m9.592s
sys     0m11.410s

[root@ac2tf01 02_code]#

```

Figure 9. Timed run without GPU

The GPU accelerated the stock prediction algorithm — which takes almost three times longer without GPU.

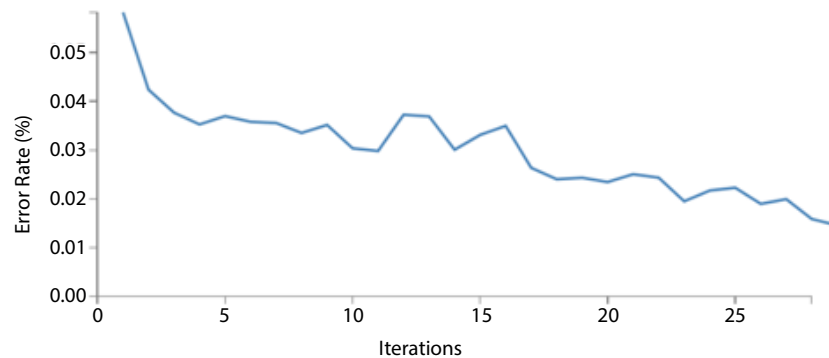


Figure 10. Stock price prediction model error rate reduction

Use Case #3: Closed Captioning

The Objective

This machine learning task was chosen to demonstrate the superiority of a multiple graphics card running in a VMWare high-performance server when compared to traditional CPUs. For this task, the team developed a large captioning model that gave imputed images a caption based on what it understood in the scene.

The algorithm trained the network for 150 epochs with 20,000 individual training steps per epoch to maximum the predictive capability of the network.

The Data

The model used data that was pulled from publicly labeled image sets to retrain a network that was originally trained to recognize images from the ImageNet® dataset.

The Algorithm

The model used the power of distributed computing to train a very large network — work that won the Stanford ImageNet Challenge. The team assigned three different physical computers to perform various individual tasks and pooled the results on a single machine. Code for the model is shown in Appendix D.

The Results

After training, the model can take in any image and caption it with decently accurate captions that could pass for a human's captioning. The network even handles images that challenge a human's ability. Example output is shown below.



"A dog is running through the
."



"A horse is running through the
grass ."



"A young boy is jumping into
the air ."



"A dog is running through the
water ."

Figure 11. Examples of captions generated by the model

Conclusion

VMware & NVIDIA have been collaborating to leverage GPUs in advanced graphics and general-purpose GPU computing. NVIDIA GPUs perform with minimal overhead on vSphere environments, while still providing the benefits of virtualization, and can significantly speed up training in vSphere Machine Learning environments. For example, a GPU-enabled VMware VM processed images almost 98 percent faster than a non-GPU VM. Likewise, the stock prediction inference engine was 66 percent faster on GPU-enabled virtual machines. These use cases have shown that vSphere is a great platform for running general purpose GPU-based machine learning.

Even better, VMware lets you run and scale machine language processing or stop it when required. Best yet, you can use vSphere to run normal day-to-day operations and use the same hypervisor to extend out and answer ML queries. This allows you to get into new lines of business using one set of hardware and one IT skill set.

References

- [SR-IOV Support for Virtualization on Infiniband Clusters](#)
- [Performance of RDMA and HPC Applications in Virtual Machines using FDR InfiniBand on VMware vSphere](#)
- [Machine Learning with NVIDIA GPUs](#)
- [What is the Difference Between AI, Machine and Deep Learning?](#)
- [What is GPU Computing?](#)
- [Simple Deep Learning Model for Stock Price Prediction](#)
- [ImageNet Challenge](#)
- [What is Machine Learning](#)
- [Roundup of Machine Learning Forecasts And Market Estimates, 2018](#)
- [Machine Learning in the Cloud: How it Can Help You Right Now](#)
- [The Best Open Source Machine Learning Frameworks](#)
- [Artistic Style Transfer](#)
- [A Simple Deep Learning Model for Stock Price Prediction Using Tensorflow](#)
- [Distributed TensorFlow Tutorial](#)
- [Image Caption Generator with Deep Learning in TensorFlow](#)

Acknowledgements

Authors

- Vineeth Yeevani
- Anusha Mohan
- Srividhya Shankar

Contributors

- Mohan Potheri
- Dean Bolton

Appendix A: TensorFlow Installation in Centos 7 Virtual Machine

Install JDK 1.8. Make sure path and JAVA_HOME are correct.

```
$ git clone https://github.com/bazelbuild/bazel.git
$ git checkout tags/0.3.2
$ cd bazel
$ ./compile.sh
```

Get the appropriate version of TensorFlow using command:

```
$ wget https://github.com/tensorflow/tensorflow/archive/v1.6.0.zip
$ unzip v1.6.0.zip
$ cd tensorflow-1.6.0/
$ bazel build --config=opt --config=cuda //tensorflow/tools/pip_
package:build_pip_package ./configure
$ bazel build --config=opt --config=cuda //tensorflow/tools/pip_
package:build_pip_package
$ bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/
tensorflow_pkg
$ pip install --upgrade /tmp/tensorflow_pkg/tensorflow-1.6.0-cp27-cp27mu-
linux_x86_64.whl
```

Appendix B: Python Code for Style Transformation Use Case

```
[root@sc2tf02 second_attempt]# cat main.py
import os
import tensorflow as tf
import numpy as np
from utils.caption_generator import Caption_Generator
from utils.test import test
import argparse
parser = argparse.ArgumentParser(description='Image Captioning software')
parser.add_argument('image_path', metavar = 'image to caption path',
type=str, help='path of image to caption')
args = parser.parse_args()
image_path = args.image_path
model_path = './models/tensorflow'
```

```

vgg_path = './data/vgg16-20160129.tfmodel'
dim_embed = 256
dim_hidden = 256
dim_in = 4096
batch_size = 1
learning_rate = 0.001
momentum = 0.9
n_epochs = 25

if not os.path.exists('data/ixtoword.npy'):
    print ('You must run 1. O\reilly Training.ipynb first.')
else:
    print(tf)
    print("Checking the reset_default_graph()")
    tf.reset_default_graph()
    with open(vgg_path,'rb') as f:
        fileContent = f.read()
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(fileContent)

    images = tf.placeholder("float32", [1, 224, 224, 3])
    tf.import_graph_def(graph_def, input_map={"images":images})

    ixtoword = np.load('data/ixtoword.npy').tolist()
    n_words = len(ixtoword)
    maxlen=15
    graph = tf.get_default_graph()
    sess = tf.InteractiveSession(graph=graph)
    caption_generator = Caption_Generator(dim_in, dim_hidden, dim_embed,
    batch_size, maxlen+2, n_words)
    graph = tf.get_default_graph()
    image, generated_words = caption_generator.build_
    generator(maxlen=maxlen)
    print("Caption from Image")
    print(test(sess, image, generated_words, ixtoword, image_path, graph,
    images))

    #for image_path in image_paths:
        #test(sess,image,generated_words,ixtoword, image_path, graph, images)

```

Appendix C: Python Code for Stock Prediction Use Case

```
[root@sc2tf02 02_code]# cat stockprediction.py
# Import
import tensorflow as tf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import csv

# Import data
data = pd.read_csv('../01_data/data_stocks.csv')

# Drop date variable
data = data.drop(['DATE'], 1)

# Dimensions of dataset
n = data.shape[0]
p = data.shape[1]

# Make data a np.array
data = data.values

# Training and test data
train_start = 0
train_end = int(np.floor(0.8*n))
test_start = train_end + 1
test_end = n
data_train = data[np.arange(train_start, train_end), :]
data_test = data[np.arange(test_start, test_end), :]

# Scale data
scaler = MinMaxScaler(feature_range=(-1, 1))
scaler.fit(data_train)
data_train = scaler.transform(data_train)
data_test = scaler.transform(data_test)
```

```
# Build X and y
X_train = data_train[:, 1:]
y_train = data_train[:, 0]
X_test = data_test[:, 1:]
y_test = data_test[:, 0]

# Number of stocks in training data
n_stocks = X_train.shape[1]

# Neurons
n_neurons_1 = 1024
n_neurons_2 = 512
n_neurons_3 = 256
n_neurons_4 = 128

# Session
net = tf.InteractiveSession()

# Placeholder
X = tf.placeholder(dtype=tf.float32, shape=[None, n_stocks])
Y = tf.placeholder(dtype=tf.float32, shape=[None])

# Initializers
sigma = 1
weight_initializer = tf.variance_scaling_initializer(mode="fan_avg",
distribution="uniform", scale=sigma)
bias_initializer = tf.zeros_initializer()

# Hidden weights
W_hidden_1 = tf.Variable(weight_initializer([n_stocks, n_neurons_1]))
bias_hidden_1 = tf.Variable(bias_initializer([n_neurons_1]))
W_hidden_2 = tf.Variable(weight_initializer([n_neurons_1, n_neurons_2]))
bias_hidden_2 = tf.Variable(bias_initializer([n_neurons_2]))
W_hidden_3 = tf.Variable(weight_initializer([n_neurons_2, n_neurons_3]))
bias_hidden_3 = tf.Variable(bias_initializer([n_neurons_3]))
W_hidden_4 = tf.Variable(weight_initializer([n_neurons_3, n_neurons_4]))
bias_hidden_4 = tf.Variable(bias_initializer([n_neurons_4]))
```

```
# Output weights
W_out = tf.Variable(weight_initializer([n_neurons_4, 1]))
bias_out = tf.Variable(bias_initializer([1]))

# Hidden layer
hidden_1 = tf.nn.relu(tf.add(tf.matmul(X, W_hidden_1), bias_hidden_1))
hidden_2 = tf.nn.relu(tf.add(tf.matmul(hidden_1, W_hidden_2), bias_hidden_2))
hidden_3 = tf.nn.relu(tf.add(tf.matmul(hidden_2, W_hidden_3), bias_hidden_3))
hidden_4 = tf.nn.relu(tf.add(tf.matmul(hidden_3, W_hidden_4), bias_hidden_4))

# Output layer (transpose!)
out = tf.transpose(tf.add(tf.matmul(hidden_4, W_out), bias_out))
#out_print = tf.Print(out, [out, Y])
# Cost function
#mse = tf.reduce_mean(tf.squared_difference(out_print, Y))
mse = tf.reduce_mean(tf.squared_difference(out, Y))
# Optimizer
opt = tf.train.AdamOptimizer().minimize(mse)

# Init
net.run(tf.global_variables_initializer())

# Fit neural net
batch_size = 256
mse_train = []
mse_test = []

# Run
epochs = 10
for e in range(epochs):

    # Shuffle training data
    shuffle_indices = np.random.permutation(np.arange(len(y_train)))
    X_train = X_train[shuffle_indices]
    y_train = y_train[shuffle_indices]
```

```
# Minibatch training
for i in range(0, len(y_train) // batch_size):
    start = i * batch_size
    batch_x = X_train[start:start + batch_size]
    batch_y = y_train[start:start + batch_size]
    # Run optimizer with batch
    net.run(opt, feed_dict={X: batch_x, Y: batch_y})

# Show progress
if np.mod(i, 50) == 0:
    # MSE train and test
    mse_train.append(net.run(mse, feed_dict={X: X_train, Y: y_train}))
    mse_test.append(net.run(mse, feed_dict={X: X_test, Y: y_test}))
    # print('MSE Train: ', mse_train[-1])
    print('MSE Test: ', mse_test[-1])
    # print("----")

# Prediction
pred = net.run(out, feed_dict={X: X_test})
# print("prediction ", pred)
print('The final error is: ', mse_test[-1])
lenarr = []
count = 0
for elem in mse_test:
    lenarr.append(count)
    count = count + 1
csv_out = open('outputfile.txt', 'wb')
mywriter = csv.writer(csv_out)
rows = zip(lenarr, mse_test)
mywriter.writerows(rows)
csv_out.close
```

Appendix D: Python Code for Image Captioning Use Case

```
import os
import tensorflow as tf
import numpy as np
from utils.caption_generator import Caption_Generator
from utils.test import test
import argparse

parser = argparse.ArgumentParser(description='Image Captioning software')

parser.add_argument('image_path', metavar = 'image to caption path',
                    type=str, help='path of image to caption')
args = parser.parse_args()
image_path = args.image_path

model_path = './models/tensorflow'
vgg_path = './data/vgg16-20160129.tfmodel'

dim_embed = 256
dim_hidden = 256
dim_in = 4096
batch_size = 1
learning_rate = 0.001
momentum = 0.9
n_epochs = 25

if not os.path.exists('data/ixtoword.npy'):
    print ('You must run 1. O\reilly Training.ipynb first.')
else:
    print(tf)
    print("Checking the reset_default_graph()")
    tf.reset_default_graph()
    with open(vgg_path,'rb') as f:
        fileContent = f.read()
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(fileContent)
```

```
images = tf.placeholder("float32", [1, 224, 224, 3])
tf.import_graph_def(graph_def, input_map={"images":images})

ixtoword = np.load('data/ixtoword.npy').tolist()
n_words = len(ixtoword)
maxlen=15
graph = tf.get_default_graph()
sess = tf.InteractiveSession(graph=graph)
caption_generator = Caption_Generator(dim_in, dim_hidden, dim_embed,
batch_size, maxlen+2, n_words)
graph = tf.get_default_graph()
image, generated_words = caption_generator.build_
generator(maxlen=maxlen)
print("Caption from Image")
print(test(sess, image, generated_words, ixtoword, image_path, graph,
images))

#for image_path in image_paths:
    #test(sess,image,generated_words,ixtoword, image_path, graph, images)
```



VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com

Copyright © 2018 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at <http://www.vmware.com/go/patents>. VMware is a registered trademark or trademark of VMware, Inc. and its subsidiaries in the United States and other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies. Item No: VMW-XXXX-XXXX_White Paper Series_MACHINE-LEARNING_0lexport.indd