WEEK 10 UPDATE - ÉMPISTOS

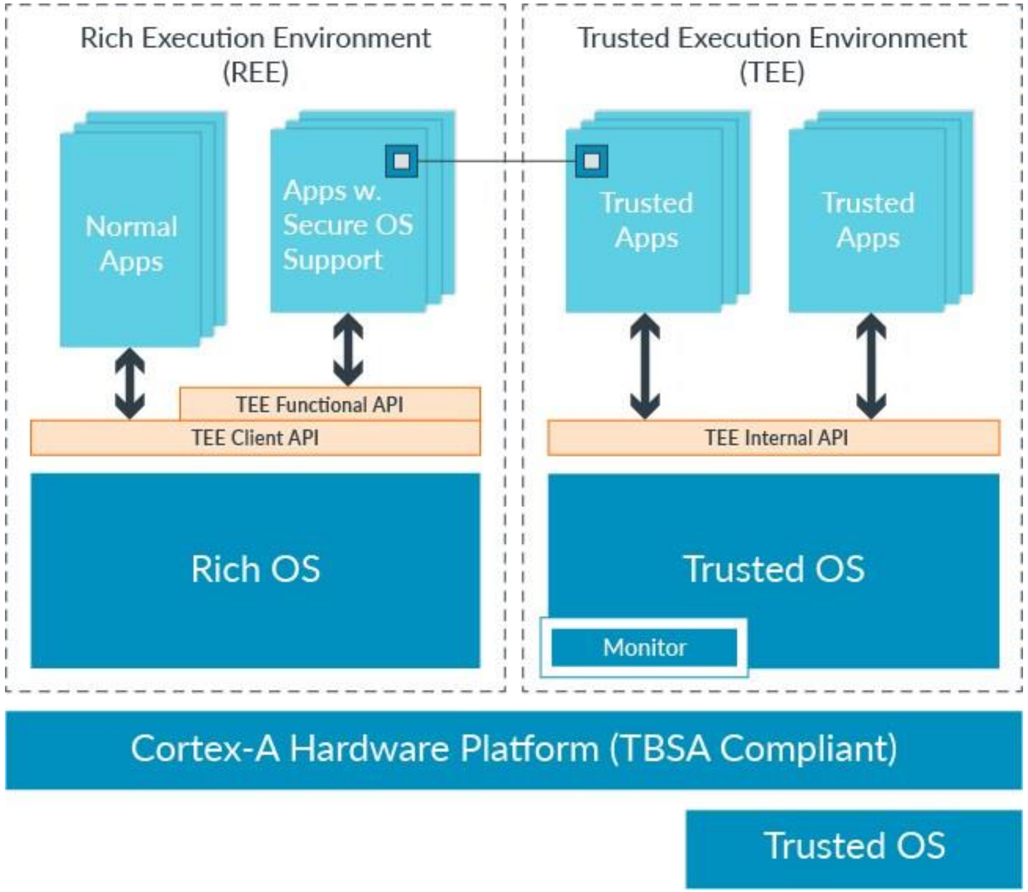
Group 14 : Max Karen, AlRaheeq AlMaktum Al Rawas, Bradford Williams, Zack Wagner, and Anay Gulati

Sponsor: Md Tanvir Arafin

GEORGE MASON UNIVERSITY

ARM TrustZone

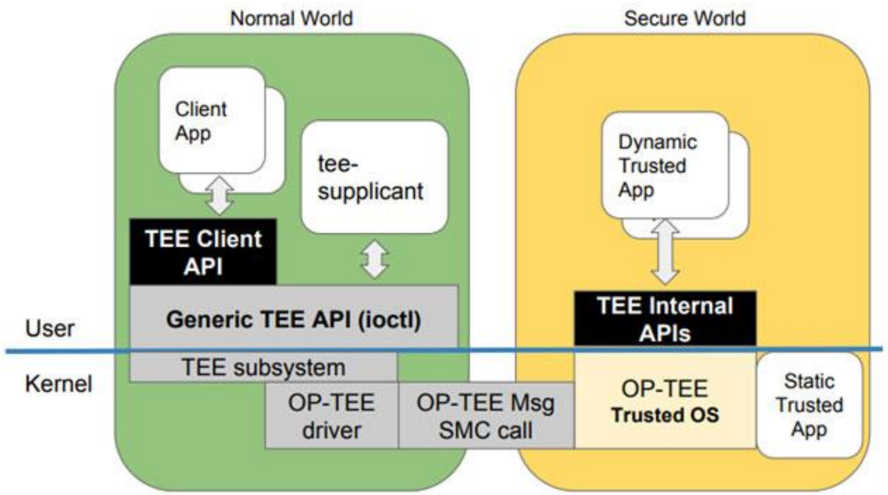
- Hardware-based security solution designed to create secure and non-secure execution environments on ARM-based processors.
- Seperates processor into Normal and Secure worlds
- Memory address space of Normal and Secure worlds through NS-bit which indicates the type of memory access
- Normal World can call Secure World through a Secure Monitor Call instruction


Rich Execution Environment (REE)
Trusted Execution Environment (TEE)
Normal Apps
Apps w. Secure OS Support
Trusted Apps
Trusted Apps
TEE Functional API
TEE Client API
TEE Internal API
Rich OS
Trusted OS
Monitor
Cortex-A Hardware Platform (TBSA Compliant)
Trusted OS

OP-TEE

"Trusted Execution Environment"

- Allows a trusted application to be ran in secure kernel world away from the non-secure OS
- Allows generic OS-level functions like Interrupt and thread handling, crypto services and shared memory
- How it works: A non-secure application calls the TEE API library, which then calls the host OS OP-TEE driver to send a request to the TEE to call a TA binary in the secure world to execute and return the result.



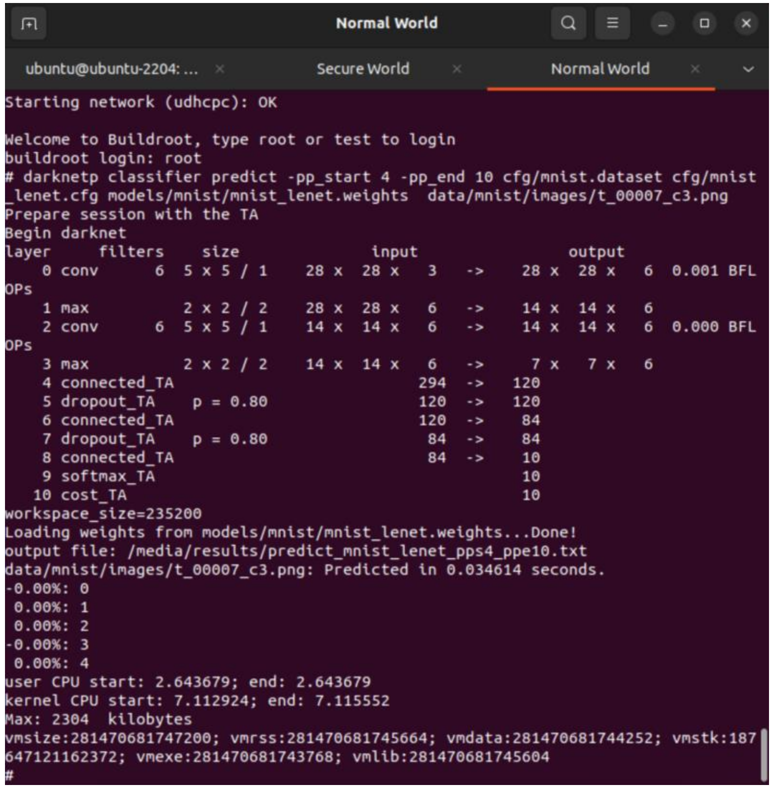
Darknet

"Open Source neural network framework written in C and CUDA"

- Optimized for speed, much faster than commonly used frameworks like TensorFlow due to reduced overhead of C programming.
- VERY Poor documentation
- Originally designed to run the YOLO object detection model

DARKNETZ

- DarkneTZ is an application that allows people to run multiple layers of a Deep Neural Network in ARM TrustZone. //Simplified version of Darknet, but is configured to take advantage of TrustZone.
- Allows for secure execution of neural network layers, particularly the final output layer, to execute in ARM TrustZone safely away from unsecure OS
    - Protects model and input data from outside adversarial attacks, such as power analysis to determine what the model is doing and poisoning the data to mess up accuracy of the model

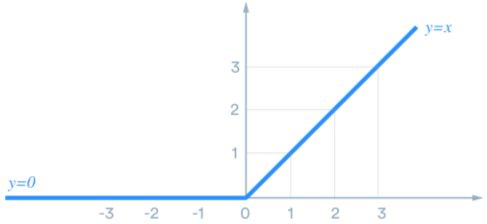


```
Starting network (udhcpc): OK

Welcome to Buildroot, type root or test to login
buildroot login: root
# darknetp classifier predict -pp_start 4 -pp_end 10 cfg/mnist.dataset cfg/mnist
_lenet.cfg models/mnist/mnist_lenet.weights  data/mnist/images/t_00007_c3.png
Prepare session with the TA
Begin darknet
layer     filters    size              input                output
    0 conv      6  5 x 5 / 1    28 x  28 x   3   ->    28 x  28 x   6  0.001 BFL
OPs
    1 max          2 x 2 / 2    28 x  28 x   6   ->    14 x  14 x   6
    2 conv      6  5 x 5 / 1    14 x  14 x   6   ->    14 x  14 x   6  0.000 BFL
OPs
    3 max          2 x 2 / 2    14 x  14 x   6   ->     7 x   7 x   6
    4 connected_TA                          294  ->   120
    5 dropout_TA    p = 0.80                120  ->   120
    6 connected_TA                          120  ->    84
    7 dropout_TA    p = 0.80                 84  ->    84
    8 connected_TA                           84  ->    10
    9 softmax_TA                                       10
   10 cost_TA                                          10
workspace_size=235200
Loading weights from models/mnist/mnist_lenet.weights...Done!
output file: /media/results/predict_mnist_lenet_pps4_ppe10.txt
data/mnist/images/t_00007_c3.png: Predicted in 0.034614 seconds.
-0.00%: 0
 0.00%: 1
 0.00%: 2
-0.00%: 3
 0.00%: 4
user CPU start: 2.643679; end: 2.643679
kernel CPU start: 7.112924; end: 7.115552
Max: 2304  kilobytes
vmsize:281470681747200; vmrss:281470681745664; vmdata:281470681744252; vmstk:187
647121162372; vmexe:281470681743768; vmlib:281470681745604
#
```

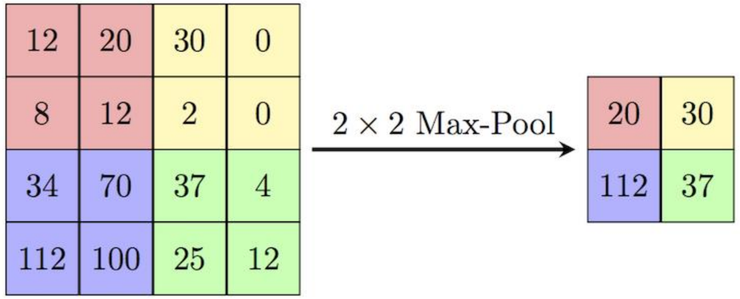
Deep Learning

- Machine based learning on artificial neural networks to recognize complex patterns
- Made from layers of ‘neurons’ which learn to recognize patterns and predict/classify things in an image
- DarkneTZ helps prevent attacks on models by having some layers of a network execute in a trusted zone

DEEP LEARNING : LAYERS

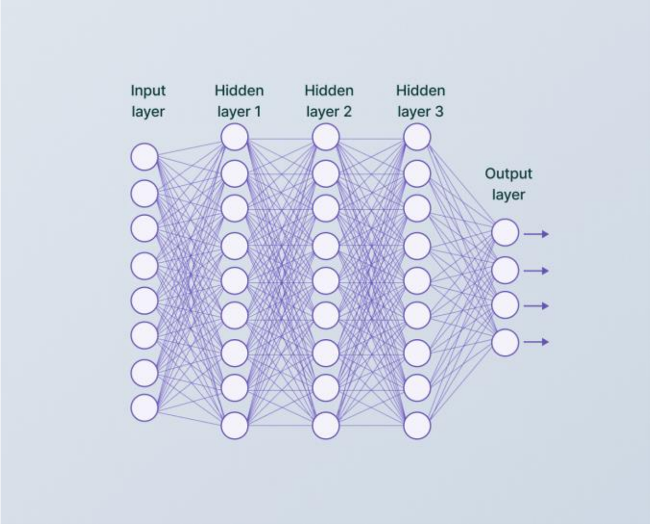


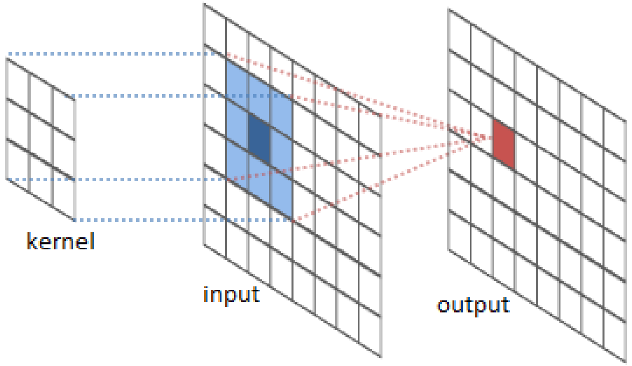
- Layers – different parts of a neural network that have different purposes
- Input Layer – input data
- Hidden Layers
  - Fully Connected layers
  - Convolutional layers
  - MaxPooling layers
- Output layer – final prediction

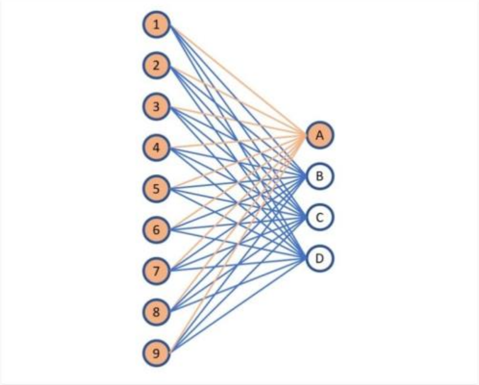


12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

20	30
112	37



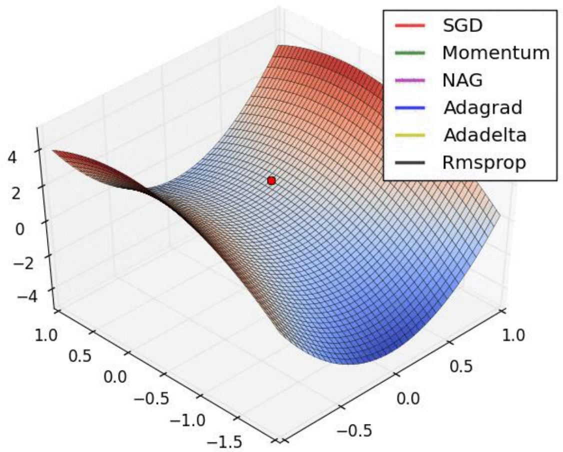




DEEP LEARNING: THE LEARNING PART

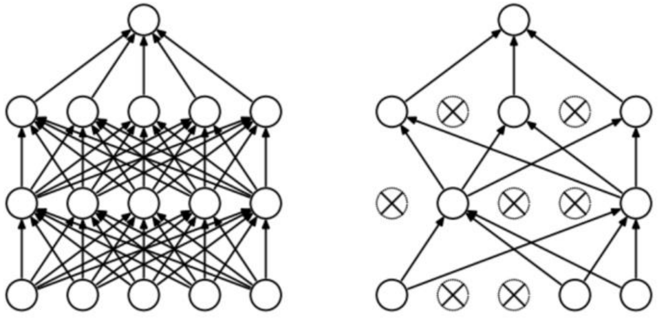
How a network learns is dependent on a few things:

- Learning rate
- Loss function
- Optimizer



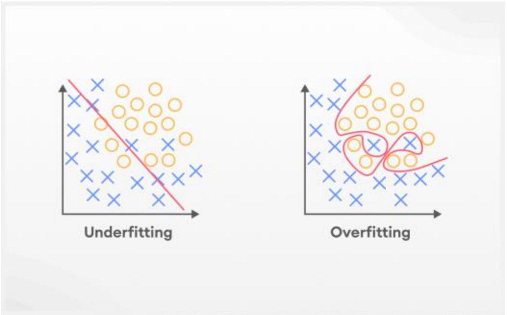
Other important terms:

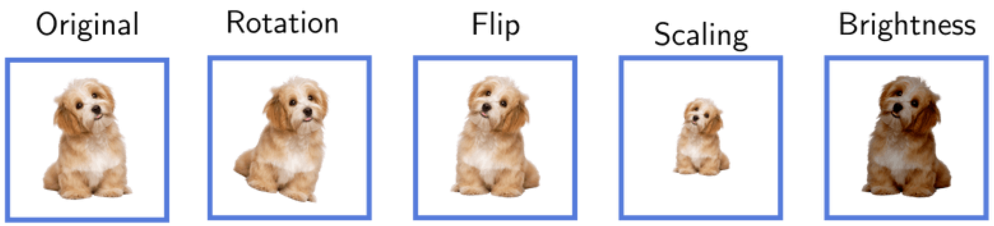
- Training vs Testing
- Underfitting vs Overfitting
- Dropout
- Data augmentation



(a) Standard Neural Net

(b) After applying dropout.



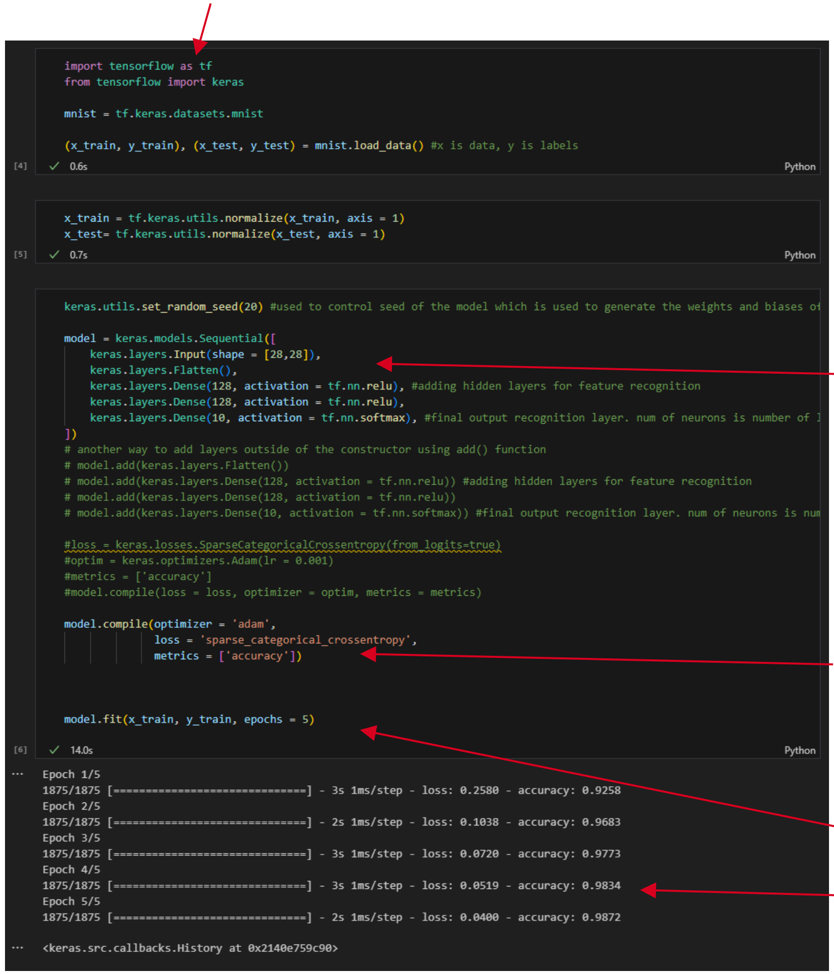


KERAS & TENSORFLOW

- Keras is an API that is bundled with the TensorFlow library that allows for easy construction, modification, and testing of neural networks
- Designed to be human-readable
- Provides very easy modules and functions to implement a neural network.

KERAS EXAMPLE – REGULAR NEURAL NETWORK

Importing tensorflow, keras, and our datasets



```
import tensorflow as tf
from tensorflow import keras

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data() #x is data, y is labels
```

[4] ✓ 0.6s Python

```
x_train = tf.keras.utils.normalize(x_train, axis = 1)
x_test= tf.keras.utils.normalize(x_test, axis = 1)
```

[5] ✓ 0.7s Python

```
keras.utils.set_random_seed(20) #used to control seed of the model which is used to generate the weights and biases of

model = keras.models.Sequential([
    keras.layers.Input(shape = [28,28]),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation = tf.nn.relu), #adding hidden layers for feature recognition
    keras.layers.Dense(128, activation = tf.nn.relu),
    keras.layers.Dense(10, activation = tf.nn.softmax), #final output recognition layer. num of neurons is number of l
])
# another way to add layers outside of the constructor using add() function
# model.add(keras.layers.Flatten())
# model.add(keras.layers.Dense(128, activation = tf.nn.relu)) #adding hidden layers for feature recognition
# model.add(keras.layers.Dense(128, activation = tf.nn.relu))
# model.add(keras.layers.Dense(10, activation = tf.nn.softmax)) #final output recognition layer. num of neurons is num

#loss = keras.losses.SparseCategoricalCrossentropy(from_logits=true)
#optim = keras.optimizers.Adam(lr = 0.001)
#metrics = ['accuracy']
#model.compile(loss = loss, optimizer = optim, metrics = metrics)

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])



model.fit(x_train, y_train, epochs = 5)
```

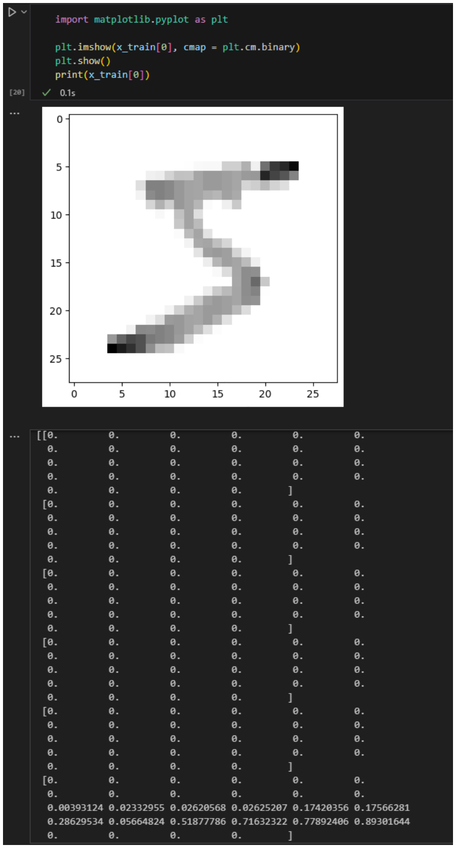
[6] ✓ 14.0s Python

```
Epoch 1/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2580 - accuracy: 0.9258
Epoch 2/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1038 - accuracy: 0.9683
Epoch 3/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0720 - accuracy: 0.9773
Epoch 4/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0519 - accuracy: 0.9834
Epoch 5/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0400 - accuracy: 0.9872

<keras.src.callbacks.History at 0x2140e759c90>
```

Creating neural network

Adding the optimizer and loss function

Training the model

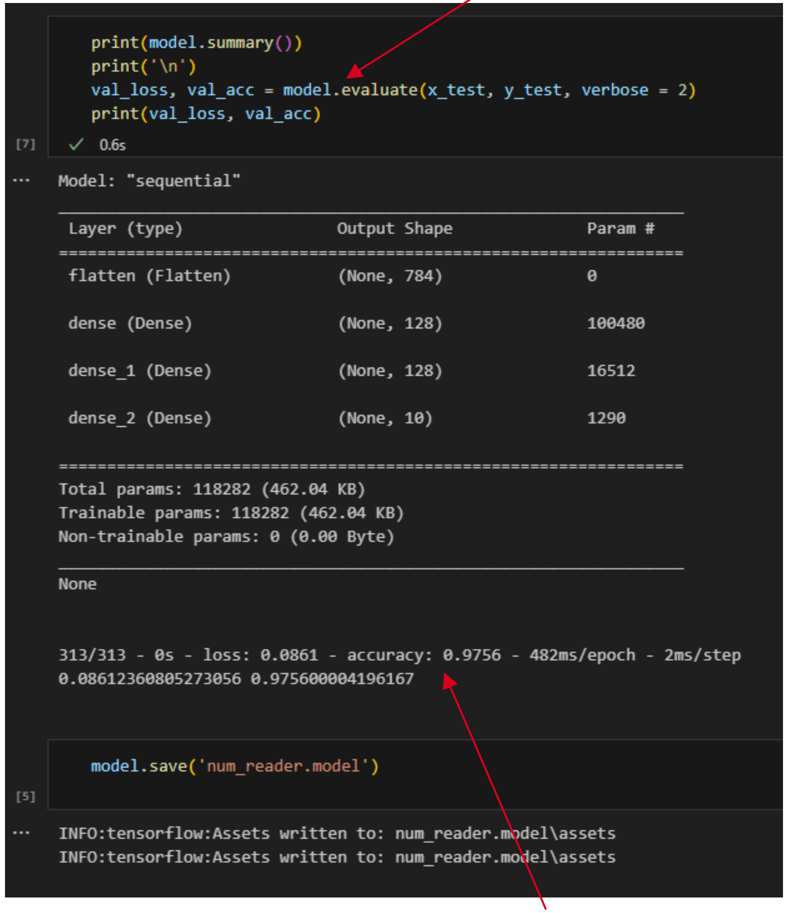


```
import matplotlib.pyplot as plt

plt.imshow(x_train[0], cmap = plt.cm.binary)
plt.show()
print(x_train[0])
```

[20] ✓ 0.1s

```
[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.00393124 0.02332955 0.02620568 0.02625207 0.17420356 0.17566281
  0.28629534 0.05664824 0.51877786 0.71632322 0.77892406 0.89301644
  0.         0.         0.         0.        ]
```

Testing the model



```
print(model.summary())
print('\n')
val_loss, val_acc = model.evaluate(x_test, y_test, verbose = 2)
print(val_loss, val_acc)
```

[7] ✓ 0.6s

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 118282 (462.04 KB)
Trainable params: 118282 (462.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None


313/313 - 0s - loss: 0.0861 - accuracy: 0.9756 - 482ms/epoch - 2ms/step
0.08612360805273056 0.975600004196167
```

```
model.save('num_reader.model')
```

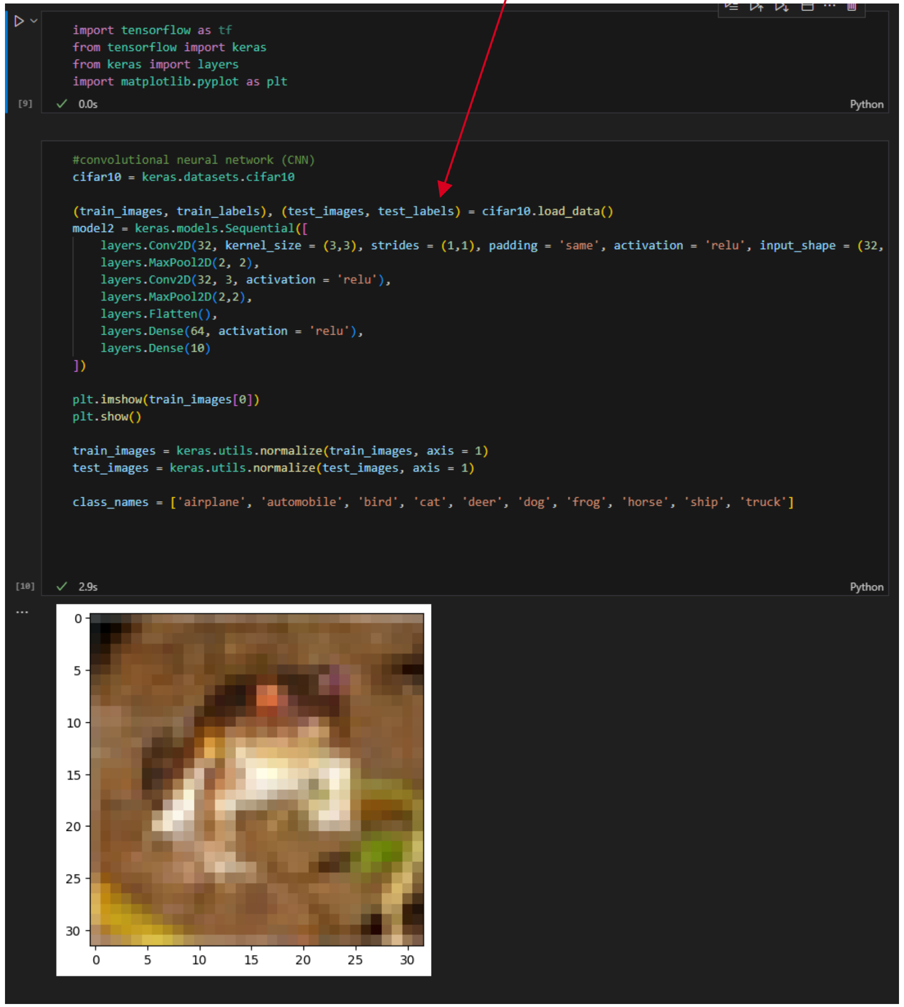
[5]

```
INFO:tensorflow:Assets written to: num_reader.model\assets
INFO:tensorflow:Assets written to: num_reader.model\assets
```

Resulting loss and accuracy

Keras Example – Convolutional Neural Network

Creating convolutional neural network

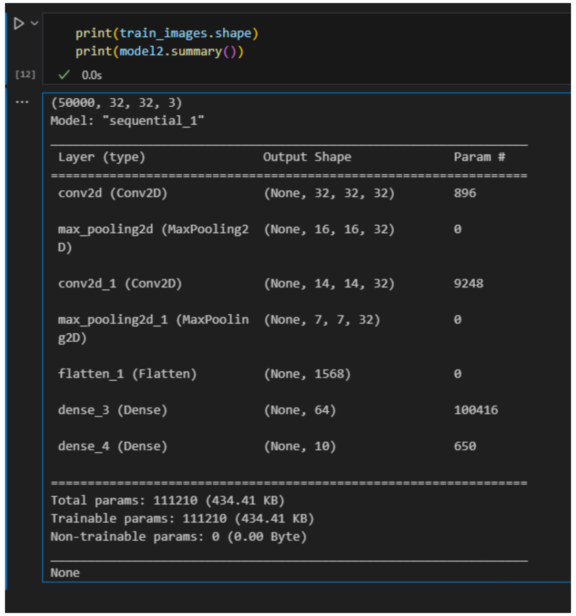


```python
import tensorflow as tf
from tensorflow import keras
from keras import layers
import matplotlib.pyplot as plt
```

```python
#convolutional neural network (CNN)
cifar10 = keras.datasets.cifar10

(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
model2 = keras.models.Sequential([
    layers.Conv2D(32, kernel_size = (3,3), strides = (1,1), padding = 'same', activation = 'relu', input_shape = (32,
    layers.MaxPool2D(2, 2),
    layers.Conv2D(32, 3, activation = 'relu'),
    layers.MaxPool2D(2,2),
    layers.Flatten(),
    layers.Dense(64, activation = 'relu'),
    layers.Dense(10)
])

plt.imshow(train_images[0])
plt.show()

train_images = keras.utils.normalize(train_images, axis = 1)
test_images = keras.utils.normalize(test_images, axis = 1)

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

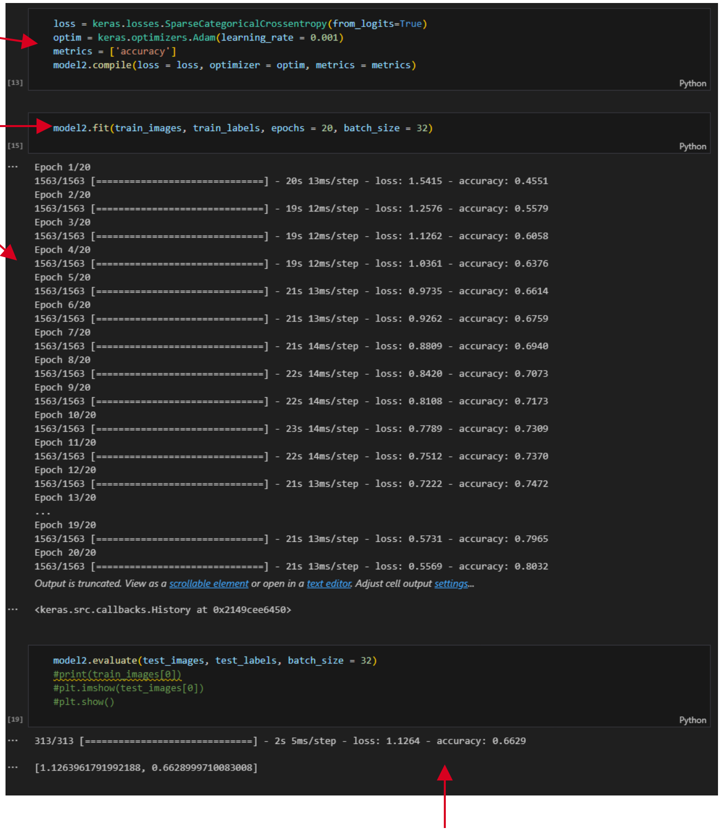
Adding optimizer and loss function

training



```python
print(train_images.shape)
print(model2.summary())
```

```
(50000, 32, 32, 3)
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 14, 14, 32)        9248

 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 32)          0
 g2D)

 flatten_1 (Flatten)         (None, 1568)              0

 dense_3 (Dense)             (None, 64)                100416

 dense_4 (Dense)             (None, 10)                650

=================================================================
Total params: 111210 (434.41 KB)
Trainable params: 111210 (434.41 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```



```python
loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True)
optim = keras.optimizers.Adam(learning_rate = 0.001)
metrics = ['accuracy']
model2.compile(loss = loss, optimizer = optim, metrics = metrics)
```

```python
model2.fit(train_images, train_labels, epochs = 20, batch_size = 32)
```

```
Epoch 1/20
1563/1563 [==============================] - 20s 13ms/step - loss: 1.5415 - accuracy: 0.4551
Epoch 2/20
1563/1563 [==============================] - 19s 12ms/step - loss: 1.2576 - accuracy: 0.5579
Epoch 3/20
1563/1563 [==============================] - 19s 12ms/step - loss: 1.1262 - accuracy: 0.6058
Epoch 4/20
1563/1563 [==============================] - 19s 12ms/step - loss: 1.0361 - accuracy: 0.6376
Epoch 5/20
1563/1563 [==============================] - 21s 13ms/step - loss: 0.9735 - accuracy: 0.6614
Epoch 6/20
1563/1563 [==============================] - 21s 13ms/step - loss: 0.9262 - accuracy: 0.6759
Epoch 7/20
1563/1563 [==============================] - 21s 14ms/step - loss: 0.8809 - accuracy: 0.6940
Epoch 8/20
1563/1563 [==============================] - 22s 14ms/step - loss: 0.8420 - accuracy: 0.7073
Epoch 9/20
1563/1563 [==============================] - 22s 14ms/step - loss: 0.8108 - accuracy: 0.7173
Epoch 10/20
1563/1563 [==============================] - 23s 14ms/step - loss: 0.7789 - accuracy: 0.7309
Epoch 11/20
1563/1563 [==============================] - 22s 14ms/step - loss: 0.7512 - accuracy: 0.7370
Epoch 12/20
1563/1563 [==============================] - 21s 13ms/step - loss: 0.7222 - accuracy: 0.7472
Epoch 13/20
...
Epoch 19/20
1563/1563 [==============================] - 21s 13ms/step - loss: 0.5731 - accuracy: 0.7965
Epoch 20/20
1563/1563 [==============================] - 21s 13ms/step - loss: 0.5569 - accuracy: 0.8032
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

<keras.src.callbacks.History at 0x2149cee6450>
```

```python
model2.evaluate(test_images, test_labels, batch_size = 32)
#print(train_images[0])
#plt.imshow(test_images[0])
#plt.show()
```

```
313/313 [==============================] - 2s 5ms/step - loss: 1.1264 - accuracy: 0.6629

[1.1263961791992188, 0.6628999710083008]
```

Testing the model

GOAL GOING FORWARD

- Learn from Keras and implement an easy-to-use Python frontend to construct and train neural networks in C that can then be securely executed using DarkneTZ
- Demonstrate basic models functioning within DarkneTZ on the Raspberry Pi 3B+
- Next task: examine darknet code and learn how to create a basic network with c instead of tensorflow

WEEK 9 - SETTING UP DARKNETZ

SETTING UP DARKNETZ ON **RASPBERRY PI 3B+**
A REPRODUCIBLE GUIDE + ISSUES

HARDWARE & SOFTWARE REQUIREMENTS

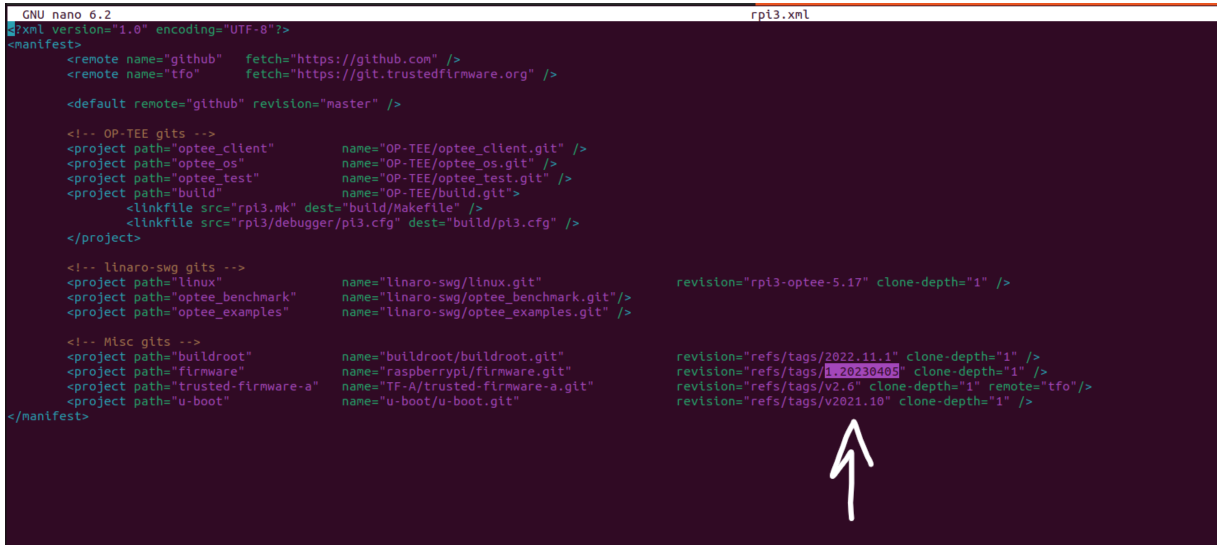
- Raspberry Pi
  - Model: 3B+
- microSD card
- USB 3.0 microSD reader
- USB to TTL Serial 3.3V Logic (UART cable)
- 5V Micro USB power supply (>2.1 A)
- Ubuntu 22.04
  - Over 50GB Hard Drive space (Building takes significant space)

BUILDING OPTEE

- Prepare build environment
  - https://optee.readthedocs.io/en/latest/building/prerequisites.html#prerequisites
  - Install required packages from apt
    - `apt install -y adb acpica-tools autoconf automake bc bison build-essential ccache cpio cscope curl device-tree-compiler e2tools expect fastboot flex ftp-upload gdisk git libattr1-dev libcap-ng-dev libfdt-dev libftdi-dev libglib2.0-dev libgmp3-dev libhidapi-dev libmpc-dev libncurses5-dev libpixman-1-dev libslirp-dev libssl-dev libtool libusb-1.0-0-dev make mtools netcat ninja-build python3-cryptography python3-pip python3-pyelftools python3-serial python-is-python3 rsync swig unzip uuid-dev wget xdg-utils xterm xz-utils zlib1g-dev`
  - Install the repo tool
    - `curl https://storage.googleapis.com/git-repo-downloads/repo > /bin/repo && chmod a+x /bin/repo`
  - Install required sub-repos with repo
    - `repo init -u https://github.com/OP-TEE/manifest.git -m rpi3.xml && repo sync`

BUILDING OPTEE

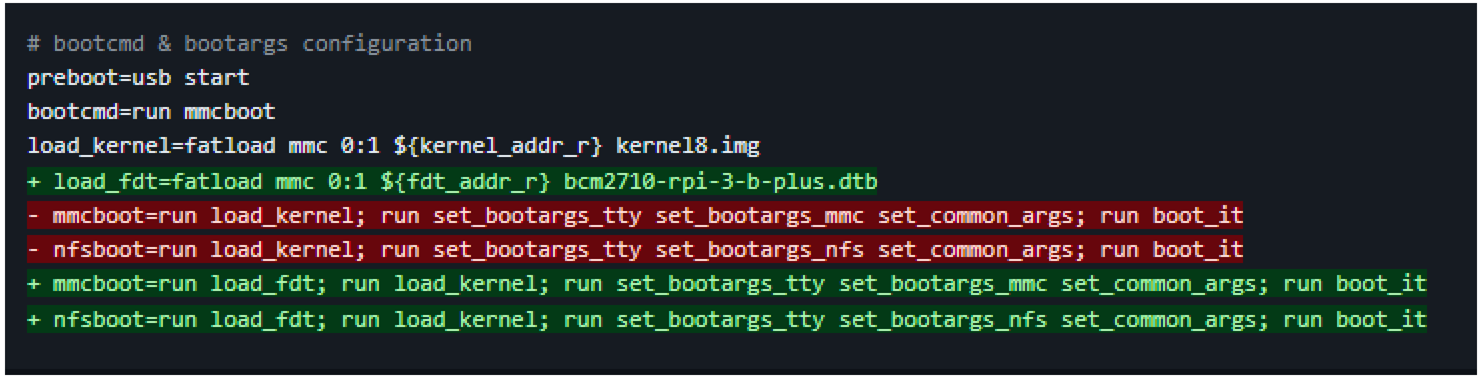
- Patch for Rasberry Pi 3B+ Hardware
  - https://github.com/OP-TEE/optee_os/issues/6284
  - Update Raspberry Pi firmware version in build manifest
    - `sed -i "s/1.20190401/1.20230405/" ./.repo/manifests/rpi3.xml`
    - Or manually change the line to the highlighted value in the image
  - Synchronize new files
    - `repo sync -m rpi3.xml --no-clone-bundle`



```
  GNU nano 6.2                                                                                      rpi3.xml
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
        <remote name="github"   fetch="https://github.com" />
        <remote name="tfo"      fetch="https://git.trustedfirmware.org" />

        <default remote="github" revision="master" />

        <!-- OP-TEE gits -->
        <project path="optee_client"         name="OP-TEE/optee_client.git" />
        <project path="optee_os"             name="OP-TEE/optee_os.git" />
        <project path="optee_test"           name="OP-TEE/optee_test.git" />
        <project path="build"                name="OP-TEE/build.git">
                <linkfile src="rpi3.mk" dest="build/Makefile" />
                <linkfile src="rpi3/debugger/pi3.cfg" dest="build/pi3.cfg" />
        </project>

        <!-- linaro-swg gits -->
        <project path="linux"                name="linaro-swg/linux.git"                   revision="rpi3-optee-5.17" clone-depth="1" />
        <project path="optee_benchmark"      name="linaro-swg/optee_benchmark.git"/>
        <project path="optee_examples"       name="linaro-swg/optee_examples.git" />

        <!-- Misc gits -->
        <project path="buildroot"            name="buildroot/buildroot.git"                revision="refs/tags/2022.11.1" clone-depth="1" />
        <project path="firmware"             name="raspberrypi/firmware.git"               revision="refs/tags/1.20230405" clone-depth="1" />
        <project path="trusted-firmware-a"   name="TF-A/trusted-firmware-a.git"            revision="refs/tags/v2.6" clone-depth="1" remote="tfo"/>
        <project path="u-boot"               name="u-boot/u-boot.git"                      revision="refs/tags/v2021.10" clone-depth="1" />
</manifest>
```

BUILDING OPTEE

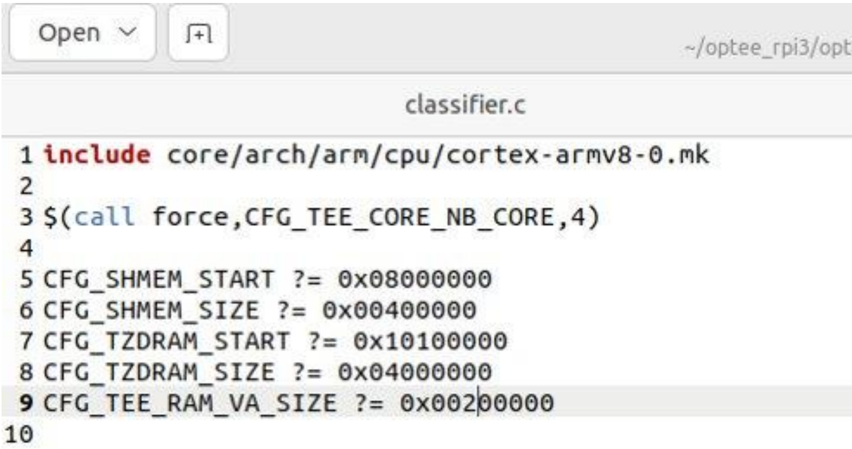
- Modify U-Boot Configuration
  - In `./build/rpi3/firmware/uboot.env.txt`
  - Add these lines
    - `load_fdt=fatload mmc 0:1 ${fdt_addr_r} bcm2710-rpi-3-b-plus.dtb`
    - `mmcboot=run load_fdt; run load_kernel; run set_bootargs_tty set_bootargs_mmc set_common_args; run boot_it`
    - `nfsboot=run load_fdt; run load_kernel; run set_bootargs_tty set_bootargs_nfs set_common_args; run boot_it`
  - Remove these lines
    - `mmcboot=run load_kernel; run set_bootargs_tty set_bootargs_mmc set_common_args; run boot_it`
    - `nfsboot=run load_kernel; run set_bootargs_tty set_bootargs_nfs set_common_args; run boot_it`



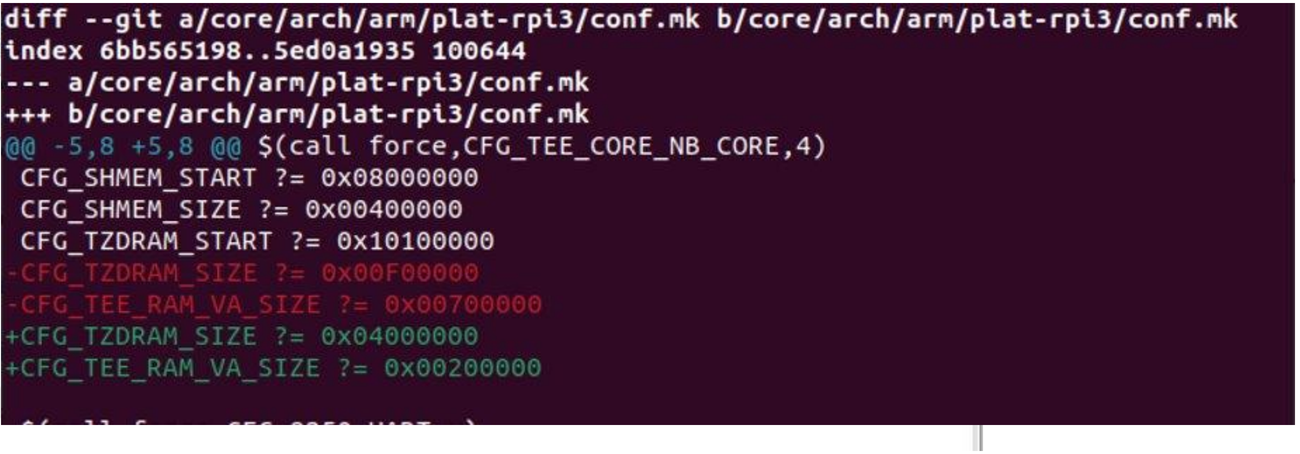
```
# bootcmd & bootargs configuration
preboot=usb start
bootcmd=run mmcboot
load_kernel=fatload mmc 0:1 ${kernel_addr_r} kernel8.img
+ load_fdt=fatload mmc 0:1 ${fdt_addr_r} bcm2710-rpi-3-b-plus.dtb
- mmcboot=run load_kernel; run set_bootargs_tty set_bootargs_mmc set_common_args; run boot_it
- nfsboot=run load_kernel; run set_bootargs_tty set_bootargs_nfs set_common_args; run boot_it
+ mmcboot=run load_fdt; run load_kernel; run set_bootargs_tty set_bootargs_mmc set_common_args; run boot_it
+ nfsboot=run load_fdt; run load_kernel; run set_bootargs_tty set_bootargs_nfs set_common_args; run boot_it
```

BUILDING OPTEE

- Increase TrustZone memory allocation
  - In `./optee_os/core/arch/arm/plat-rpi3/conf.mk`
  - Add the lines pictured below, file should look like the left image



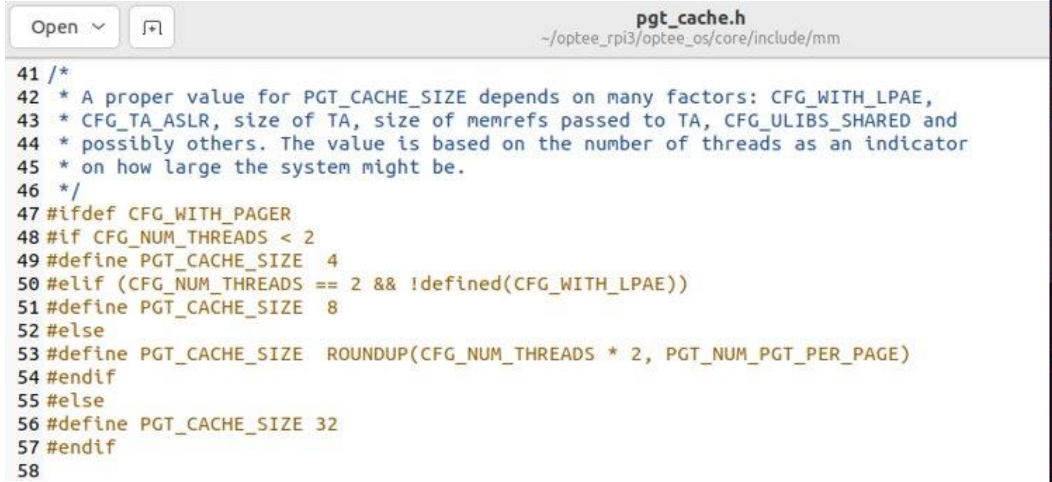
```
include core/arch/arm/cpu/cortex-armv8-0.mk

$(call force,CFG_TEE_CORE_NB_CORE,4)

CFG_SHMEM_START ?= 0x08000000
CFG_SHMEM_SIZE ?= 0x00400000
CFG_TZDRAM_START ?= 0x10100000
CFG_TZDRAM_SIZE ?= 0x04000000
CFG_TEE_RAM_VA_SIZE ?= 0x00200000

```



```
diff --git a/core/arch/arm/plat-rpi3/conf.mk b/core/arch/arm/plat-rpi3/conf.mk
index 6bb565198..5ed0a1935 100644
--- a/core/arch/arm/plat-rpi3/conf.mk
+++ b/core/arch/arm/plat-rpi3/conf.mk
@@ -5,8 +5,8 @@ $(call force,CFG_TEE_CORE_NB_CORE,4)
 CFG_SHMEM_START ?= 0x08000000
 CFG_SHMEM_SIZE ?= 0x00400000
 CFG_TZDRAM_START ?= 0x10100000
-CFG_TZDRAM_SIZE ?= 0x00F00000
-CFG_TEE_RAM_VA_SIZE ?= 0x00700000
+CFG_TZDRAM_SIZE ?= 0x04000000
+CFG_TEE_RAM_VA_SIZE ?= 0x00200000
```

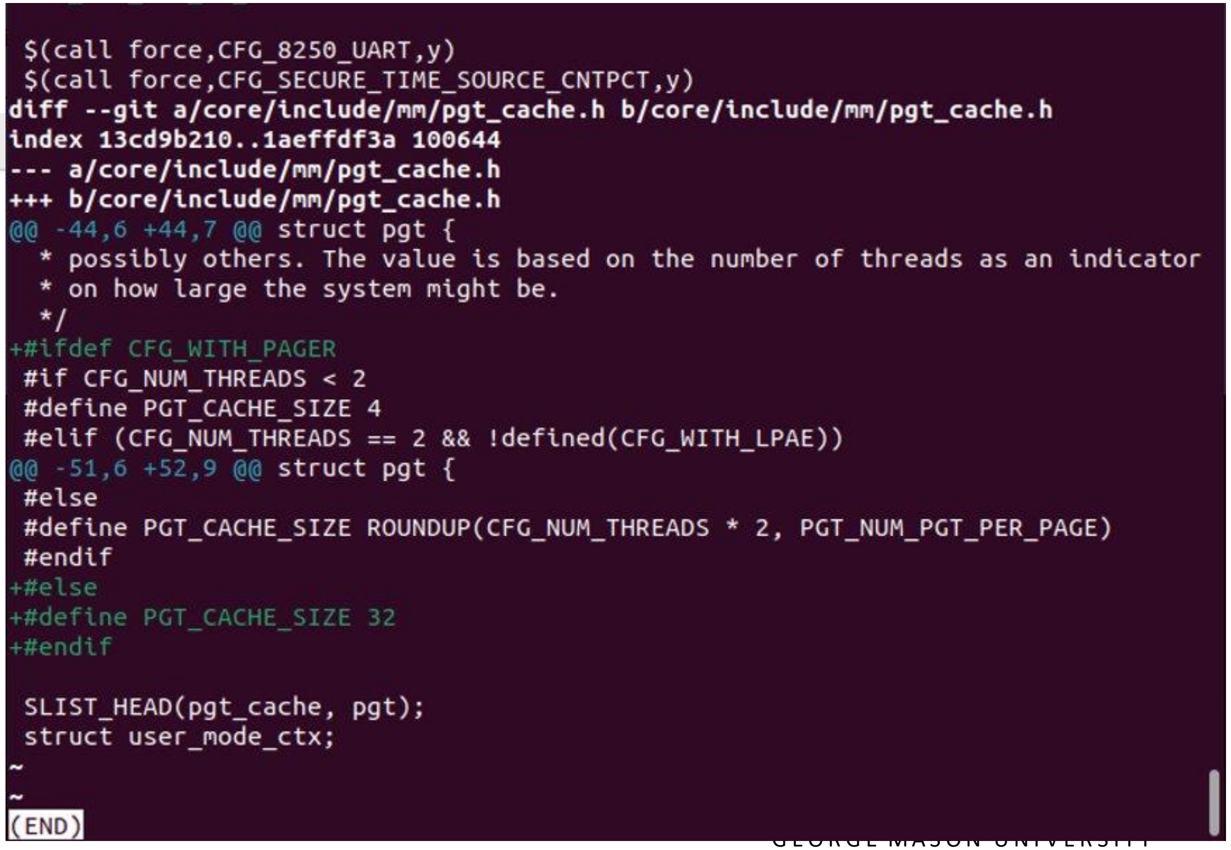
BUILDING OPTEE

- Increase page table cache size
  - In `./optee_os/core/include/mm/pgt_cache.h`
  - Add the lines pictured below, file should look like the left image



pgt_cache.h
~/optee_rpi3/optee_os/core/include/mm

```
/*
 * A proper value for PGT_CACHE_SIZE depends on many factors: CFG_WITH_LPAE,
 * CFG_TA_ASLR, size of TA, size of memrefs passed to TA, CFG_ULIBS_SHARED and
 * possibly others. The value is based on the number of threads as an indicator
 * on how large the system might be.
 */
#ifdef CFG_WITH_PAGER
#if CFG_NUM_THREADS < 2
#define PGT_CACHE_SIZE  4
#elif (CFG_NUM_THREADS == 2 && !defined(CFG_WITH_LPAE))
#define PGT_CACHE_SIZE  8
#else
#define PGT_CACHE_SIZE  ROUNDUP(CFG_NUM_THREADS * 2, PGT_NUM_PGT_PER_PAGE)
#endif
#else
#define PGT_CACHE_SIZE 32
#endif

```



```
 $(call force,CFG_8250_UART,y)
 $(call force,CFG_SECURE_TIME_SOURCE_CNTPCT,y)
diff --git a/core/include/mm/pgt_cache.h b/core/include/mm/pgt_cache.h
index 13cd9b210..1aeffdf3a 100644
--- a/core/include/mm/pgt_cache.h
+++ b/core/include/mm/pgt_cache.h
@@ -44,6 +44,7 @@ struct pgt {
  * possibly others. The value is based on the number of threads as an indicator
  * on how large the system might be.
  */
+#ifdef CFG_WITH_PAGER
 #if CFG_NUM_THREADS < 2
 #define PGT_CACHE_SIZE 4
 #elif (CFG_NUM_THREADS == 2 && !defined(CFG_WITH_LPAE))
@@ -51,6 +52,9 @@ struct pgt {
 #else
 #define PGT_CACHE_SIZE ROUNDUP(CFG_NUM_THREADS * 2, PGT_NUM_PGT_PER_PAGE)
 #endif
+#else
+#define PGT_CACHE_SIZE 32
+#endif

 SLIST_HEAD(pgt_cache, pgt);
 struct user_mode_ctx;
~
~
(END)
```

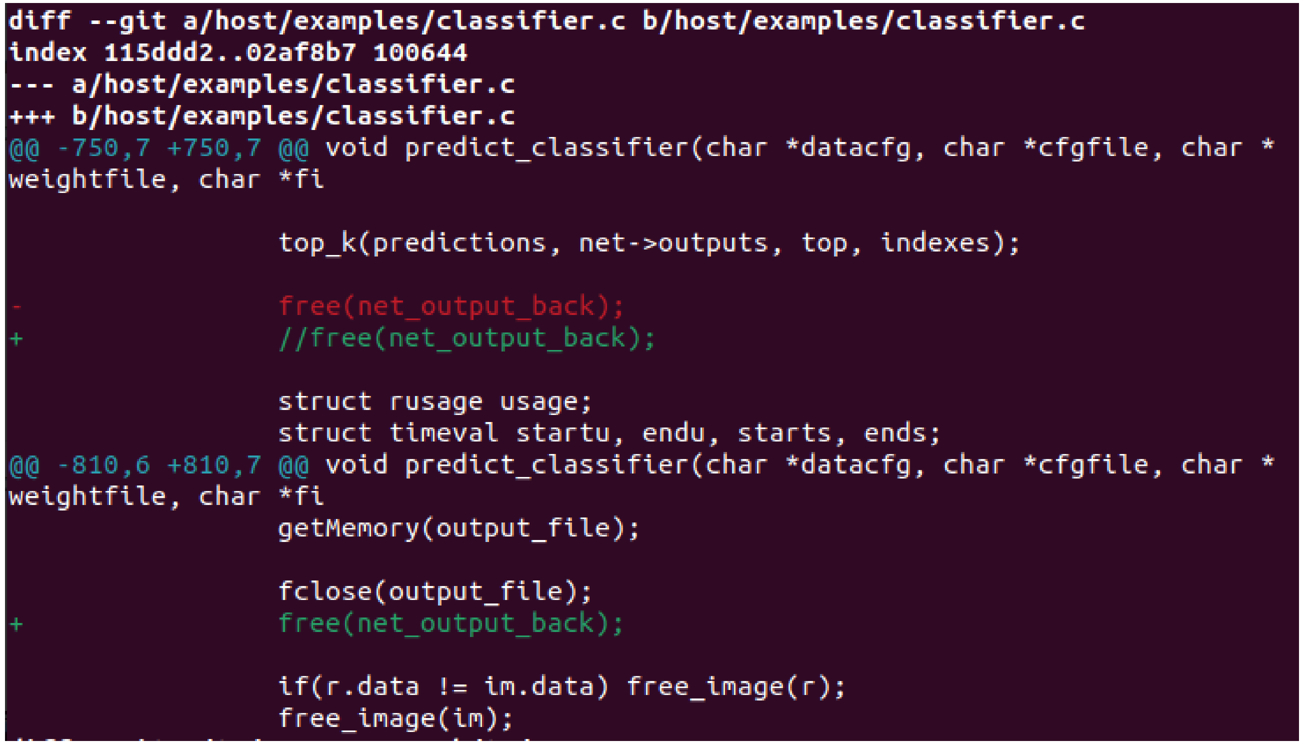
BUILDING OPTEE – ADDING DARKNETZ

https://github.com/mofanv/darknetz#2-build-darknetz

- Clone code and datasets:
  - `git clone https://github.com/mofanv/darknetz.git`
  - `git clone https://github.com/mofanv/tz_datasets.git`
- Copy into OPTEE folder
  - `mkdir $PATH_OPTEE$/optee_examples/darknetz`
  - `cp -a $PATH_darknetz/. $PATH_OPTEE/optee_examples/darknetz/`
  - `cp -a $PATH_tz_datasets/. $PATH_OPTEE/out-br/target/root/`

Building optee – Adding DarknetZ

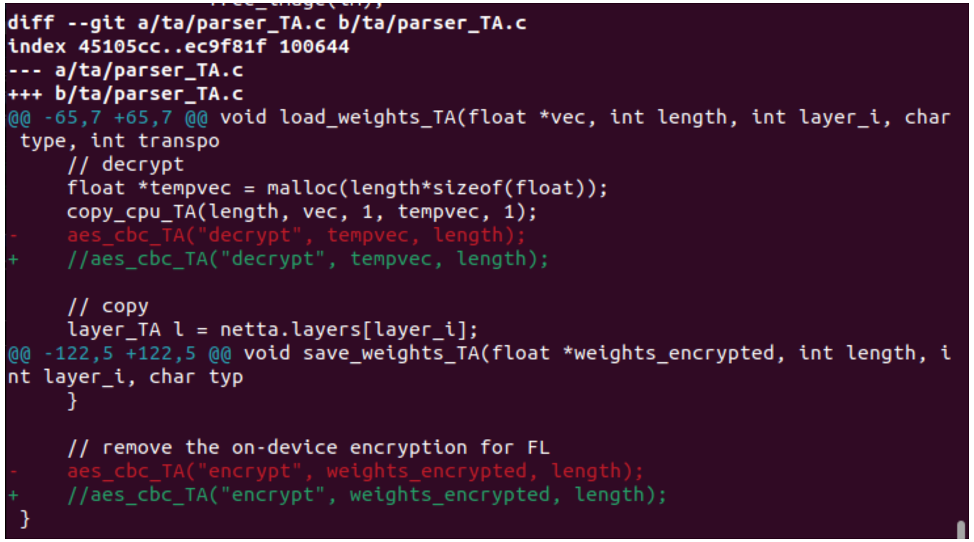
- Fix use after free issue
  - In `./optee_examples/darknetz/host/examples/classifer.c`
    - Move free(net_output_back) to the end of the file as shown



```
diff --git a/host/examples/classifier.c b/host/examples/classifier.c
index 115ddd2..02af8b7 100644
--- a/host/examples/classifier.c
+++ b/host/examples/classifier.c
@@ -750,7 +750,7 @@ void predict_classifier(char *datacfg, char *cfgfile, char *
weightfile, char *fi

                 top_k(predictions, net->outputs, top, indexes);

-                free(net_output_back);
+                //free(net_output_back);

                 struct rusage usage;
                 struct timeval startu, endu, starts, ends;
@@ -810,6 +810,7 @@ void predict_classifier(char *datacfg, char *cfgfile, char *
weightfile, char *fi
                 getMemory(output_file);

                 fclose(output_file);
+                free(net_output_back);

                 if(r.data != im.data) free_image(r);
                 free_image(im);
```

Building OPTEE – Adding DarknetZ

- Remove weight encryption
  - This is optional, but the same layers must be run in TrustZone between training and inference if this is left on.
  - In `./optee_examples/darknetz/ta/parser_TA.c`
  - Remove aes_cbc_TA(“encrypt”, weights_encrypted, length);
    - 2 spots to remove it from, as shown below



```
diff --git a/ta/parser_TA.c b/ta/parser_TA.c
index 45105cc..ec9f81f 100644
--- a/ta/parser_TA.c
+++ b/ta/parser_TA.c
@@ -65,7 +65,7 @@ void load_weights_TA(float *vec, int length, int layer_i, char
 type, int transpo
     // decrypt
     float *tempvec = malloc(length*sizeof(float));
     copy_cpu_TA(length, vec, 1, tempvec, 1);
-    aes_cbc_TA("decrypt", tempvec, length);
+    //aes_cbc_TA("decrypt", tempvec, length);

     // copy
     layer_TA l = netta.layers[layer_i];
@@ -122,5 +122,5 @@ void save_weights_TA(float *weights_encrypted, int length, i
nt layer_i, char typ
     }

     // remove the on-device encryption for FL
-    aes_cbc_TA("encrypt", weights_encrypted, length);
+    //aes_cbc_TA("encrypt", weights_encrypted, length);
 }
```

BUILDING OPTEE

- Accquire toolchains
  - `make –j2 toolchains`
- Make OPTEE with DarknetZ included
  - `make`
- Check for completed image file
  - `ls ./out/rpi3-sdcard.img`

BUILDING OPTEE – FLASHING TO SD CARD

https://optee.readthedocs.io/en/latest/building/devices/rpi3.html

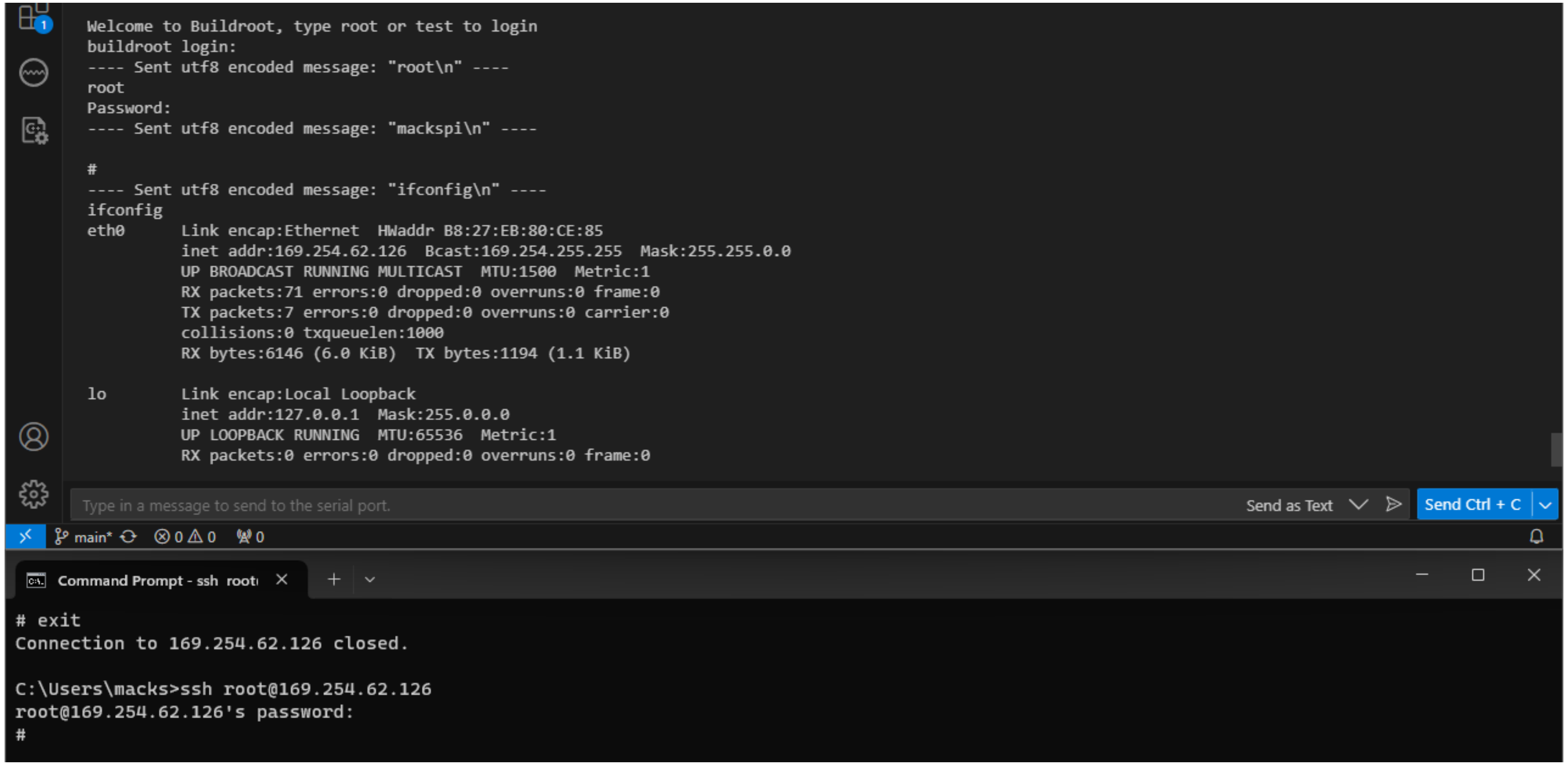
- Run ‘make img-help’
  - Find your microSD card name (dmesg or lsblk to help find the name)
  - Flash using instructions from make command
- Plug the microSD into the Raspberry Pi 3B+ board
- Run ‘picocom –b 115200 /dev/ttyUSB0’
  - Possibly install picocom with ‘apt-get install picocom’
- Turn on the Raspberry Pi board
- Run ‘xtest’

RUNNING DARKNETZ AND BOOTING RASPBERRY PI3

- After booting the Raspberry Pi, run the following command to test:
  - *Darknetp*
- The following should be outputted if it was successful:
  - *# usage: ./darknetp <function>*

Running DarknetZ and booting Raspberry pi3

- To ssh into the Raspberry pi3, go to /etc/ssh/sshd_config
  - set "*PermitEmptyPasswords yes*" to log in as test user
  - To log in as root user, additionally set *"PermitRootLogin yes"*



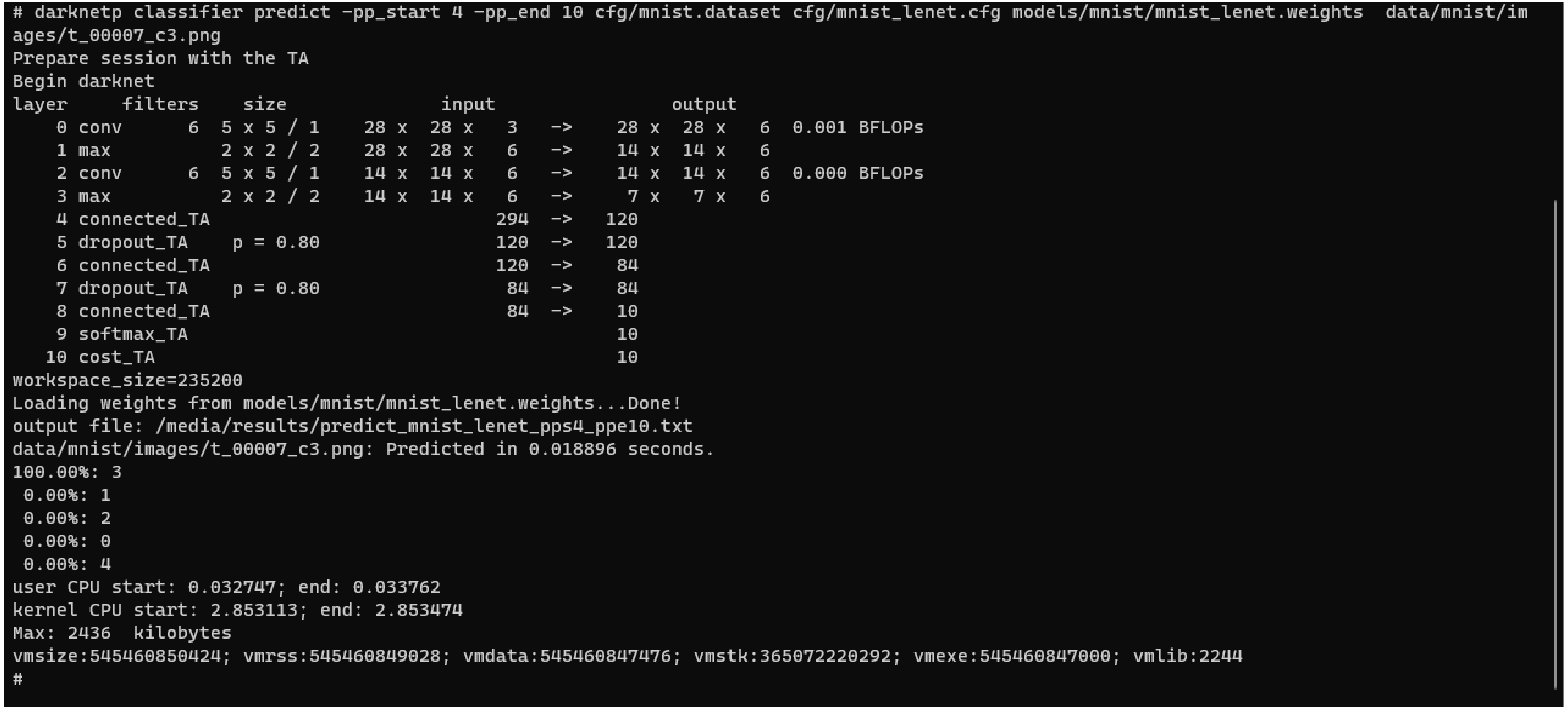
```
Welcome to Buildroot, type root or test to login
buildroot login:
---- Sent utf8 encoded message: "root\n" ----
root
Password:
---- Sent utf8 encoded message: "mackspi\n" ----

#
---- Sent utf8 encoded message: "ifconfig\n" ----
ifconfig
eth0      Link encap:Ethernet  HWaddr B8:27:EB:80:CE:85
          inet addr:169.254.62.126  Bcast:169.254.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:71 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6146 (6.0 KiB)  TX bytes:1194 (1.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

Type in a message to send to the serial port.
Send as Text
Send Ctrl + C
main*
0
0
0
Command Prompt - ssh root

```
# exit
Connection to 169.254.62.126 closed.

C:\Users\macks>ssh root@169.254.62.126
root@169.254.62.126's password:
#
```

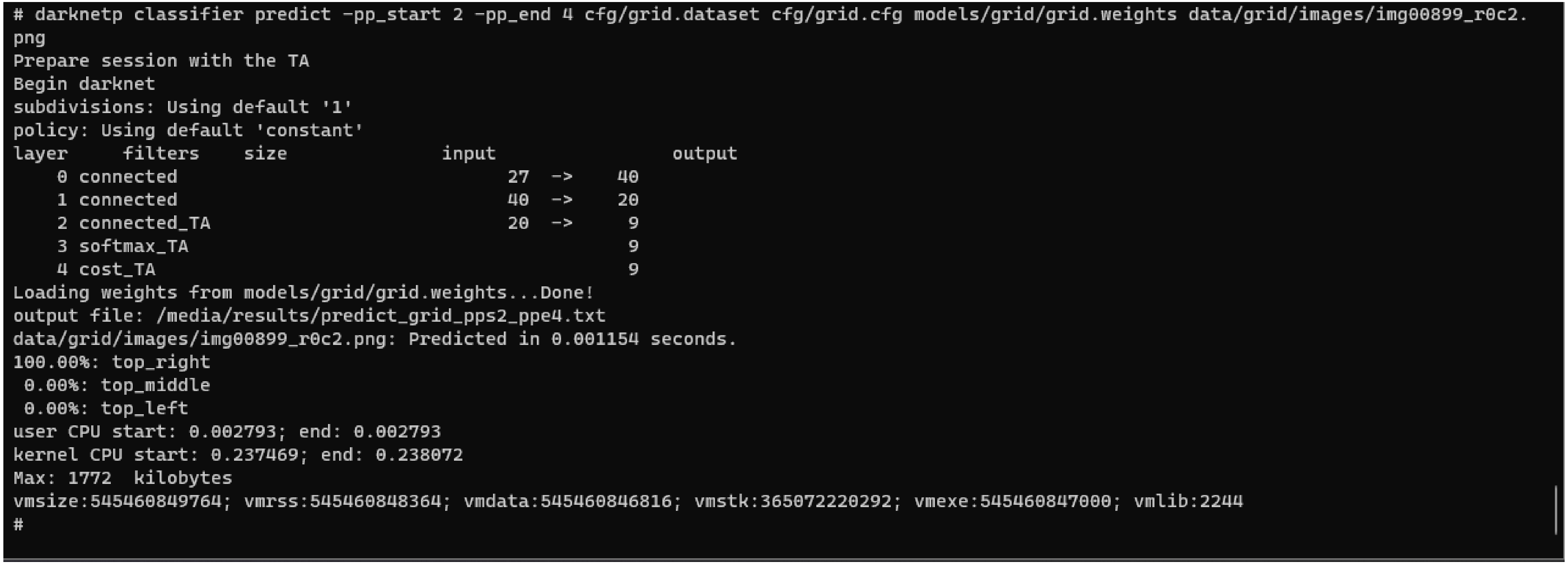
Running DarknetZ and booting raspberry pi3

- Running on MNIST Dataset



```
# darknetp classifier predict -pp_start 4 -pp_end 10 cfg/mnist.dataset cfg/mnist_lenet.cfg models/mnist/mnist_lenet.weights  data/mnist/im
ages/t_00007_c3.png
Prepare session with the TA
Begin darknet
layer     filters    size              input                output
    0 conv      6  5 x 5 / 1    28 x  28 x   3   ->    28 x  28 x   6  0.001 BFLOPs
    1 max          2 x 2 / 2    28 x  28 x   6   ->    14 x  14 x   6
    2 conv      6  5 x 5 / 1    14 x  14 x   6   ->    14 x  14 x   6  0.000 BFLOPs
    3 max          2 x 2 / 2    14 x  14 x   6   ->     7 x   7 x   6
    4 connected_TA                          294   ->   120
    5 dropout_TA    p = 0.80                120   ->   120
    6 connected_TA                          120   ->    84
    7 dropout_TA    p = 0.80                 84   ->    84
    8 connected_TA                           84   ->    10
    9 softmax_TA                                       10
   10 cost_TA                                          10
workspace_size=235200
Loading weights from models/mnist/mnist_lenet.weights...Done!
output file: /media/results/predict_mnist_lenet_pps4_ppe10.txt
data/mnist/images/t_00007_c3.png: Predicted in 0.018896 seconds.
100.00%: 3
 0.00%: 1
 0.00%: 2
 0.00%: 0
 0.00%: 4
user CPU start: 0.032747; end: 0.033762
kernel CPU start: 2.853113; end: 2.853474
Max: 2436  kilobytes
vmsize:545460850424; vmrss:545460849028; vmdata:545460847476; vmstk:365072220292; vmexe:545460847000; vmlib:2244
#
```

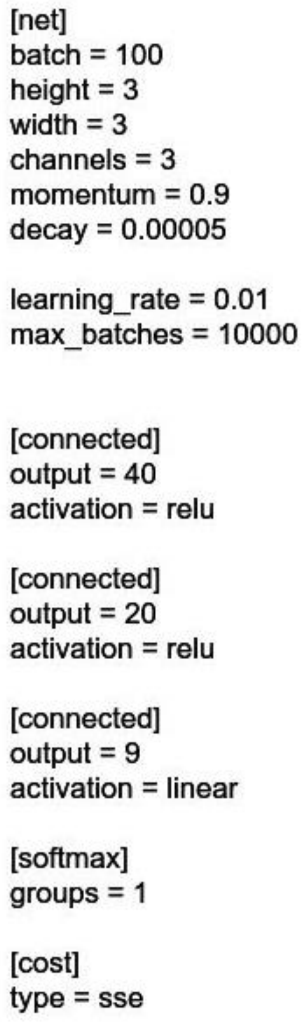
Working Example 'Grid'



```
# darknetp classifier predict -pp_start 2 -pp_end 4 cfg/grid.dataset cfg/grid.cfg models/grid/grid.weights data/grid/images/img00899_r0c2.
png
Prepare session with the TA
Begin darknet
subdivisions: Using default '1'
policy: Using default 'constant'
layer     filters    size              input               output
    0 connected                              27  ->    40
    1 connected                              40  ->    20
    2 connected_TA                           20  ->     9
    3 softmax_TA                                        9
    4 cost_TA                                           9
Loading weights from models/grid/grid.weights...Done!
output file: /media/results/predict_grid_pps2_ppe4.txt
data/grid/images/img00899_r0c2.png: Predicted in 0.001154 seconds.
100.00%: top_right
 0.00%: top_middle
 0.00%: top_left
user CPU start: 0.002793; end: 0.002793
kernel CPU start: 0.237469; end: 0.238072
Max: 1772  kilobytes
vmsize:545460849764; vmrss:545460848364; vmdata:545460846816; vmstk:365072220292; vmexe:545460847000; vmlib:2244
#
```

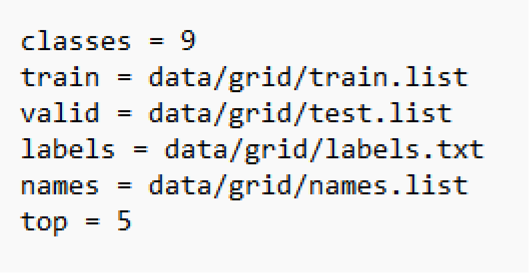
WORKING EXAMPLE 'GRID' CONT.

- grid.cfg



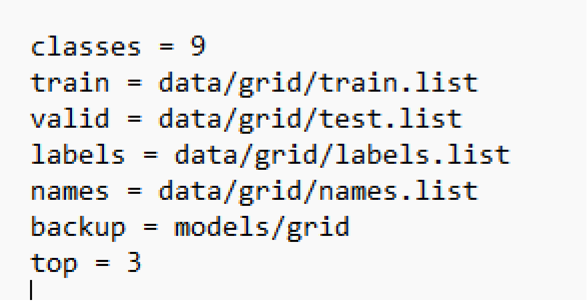
```
[net]
batch = 100
height = 3
width = 3
channels = 3
momentum = 0.9
decay = 0.00005

learning_rate = 0.01
max_batches = 10000


[connected]
output = 40
activation = relu

[connected]
output = 20
activation = relu

[connected]
output = 9
activation = linear

[softmax]
groups = 1

[cost]
type = sse
```

- grid.data



```
classes = 9
train = data/grid/train.list
valid = data/grid/test.list
labels = data/grid/labels.txt
names = data/grid/names.list
top = 5
```

- grid.dataset

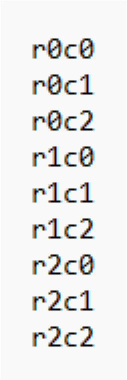


```
classes = 9
train = data/grid/train.list
valid = data/grid/test.list
labels = data/grid/labels.list
names = data/grid/names.list
backup = models/grid
top = 3
```

WORKING EXAMPLE ‘GRID’ CONT.

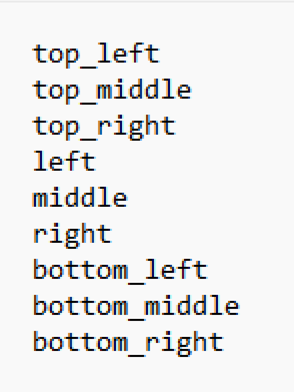
- In a new directory named ‘grid’

- labels.list



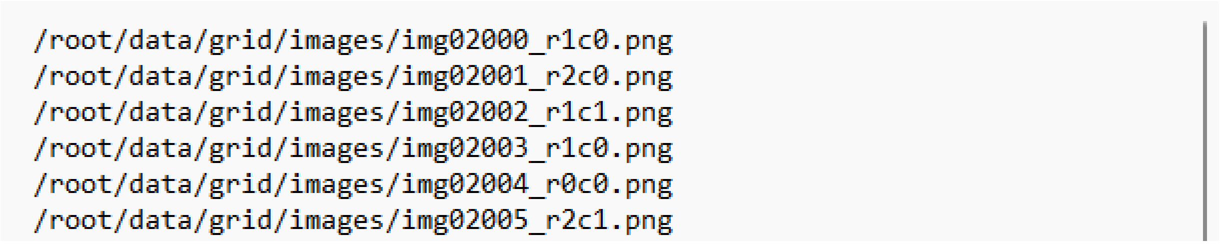
```
r0c0
r0c1
r0c2
r1c0
r1c1
r1c2
r2c0
r2c1
r2c2
```

- names.list



```
top_left
top_middle
top_right
left
middle
right
bottom_left
bottom_middle
bottom_right
```

- test.list
  - A file containing the pathways for images 2000-2099
  - Images generated on next slide
  - Example below

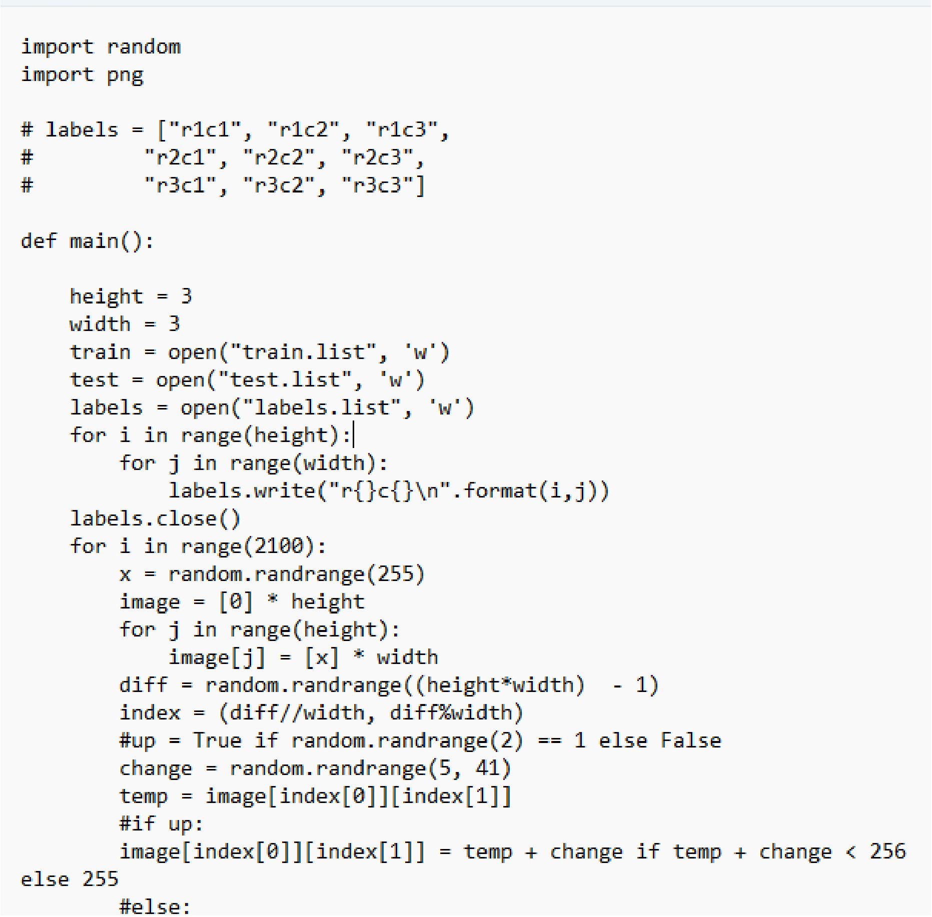


```
/root/data/grid/images/img02000_r1c0.png
/root/data/grid/images/img02001_r2c0.png
/root/data/grid/images/img02002_r1c1.png
/root/data/grid/images/img02003_r1c0.png
/root/data/grid/images/img02004_r0c0.png
/root/data/grid/images/img02005_r2c1.png
```

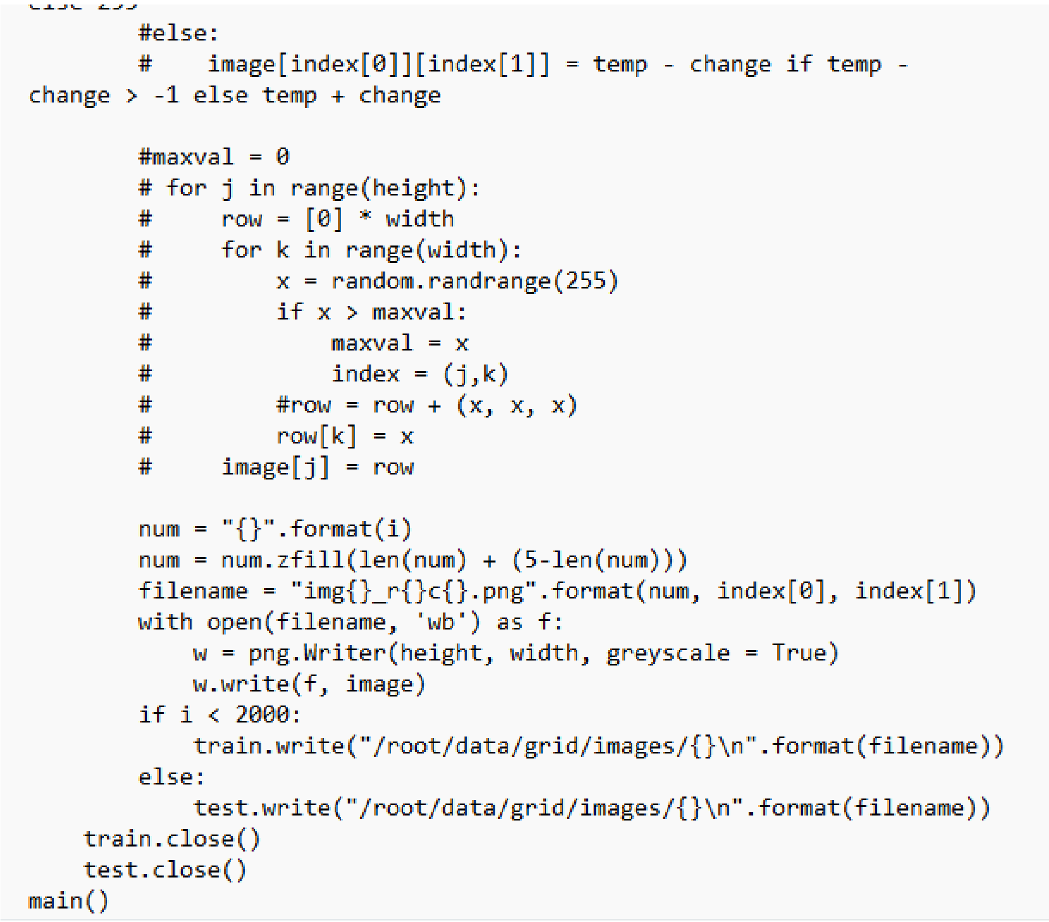
- train.list
  - A file containing the pathways for images 0000-1999
  - Images generated on next slide
  - Example to the right

Working Example 'Grid' Cont.

- In a new directory named 'images' inside the directory 'grid'
  - Create and run the following python program



```
import random
import png

# labels = ["r1c1", "r1c2", "r1c3",
#         "r2c1", "r2c2", "r2c3",
#         "r3c1", "r3c2", "r3c3"]

def main():

    height = 3
    width = 3
    train = open("train.list", 'w')
    test = open("test.list", 'w')
    labels = open("labels.list", 'w')
    for i in range(height):
        for j in range(width):
            labels.write("r{}c{}\n".format(i,j))
    labels.close()
    for i in range(2100):
        x = random.randrange(255)
        image = [0] * height
        for j in range(height):
            image[j] = [x] * width
        diff = random.randrange((height*width)  - 1)
        index = (diff//width, diff%width)
        #up = True if random.randrange(2) == 1 else False
        change = random.randrange(5, 41)
        temp = image[index[0]][index[1]]
        #if up:
        image[index[0]][index[1]] = temp + change if temp + change < 256
else 255
        #else:
```



```
        #else:
        #    image[index[0]][index[1]] = temp - change if temp -
change > -1 else temp + change

        #maxval = 0
        # for j in range(height):
        #     row = [0] * width
        #     for k in range(width):
        #         x = random.randrange(255)
        #         if x > maxval:
        #             maxval = x
        #             index = (j,k)
        #         #row = row + (x, x, x)
        #         row[k] = x
        #     image[j] = row

        num = "{}".format(i)
        num = num.zfill(len(num) + (5-len(num)))
        filename = "img{}_r{}c{}.png".format(num, index[0], index[1])
        with open(filename, 'wb') as f:
            w = png.Writer(height, width, greyscale = True)
            w.write(f, image)
        if i < 2000:
            train.write("/root/data/grid/images/{}\n".format(filename))
        else:
            test.write("/root/data/grid/images/{}\n".format(filename))
    train.close()
    test.close()
main()
```

WEEK 9 - SETTING UP DARKNETZ

SETTING UP DARKNETZ ON **QEMU**
A REPRODUCIBLE GUIDE + ISSUES

Software Requirements

- Ubuntu 22.04 or 20.04 running VM.

*Note: make sure to have sufficient VM Memory Space (Recommended: 50+ GB)

INSTALLING OPTEE

Use the following GitHub repository as the guide to install and build DarknetZ

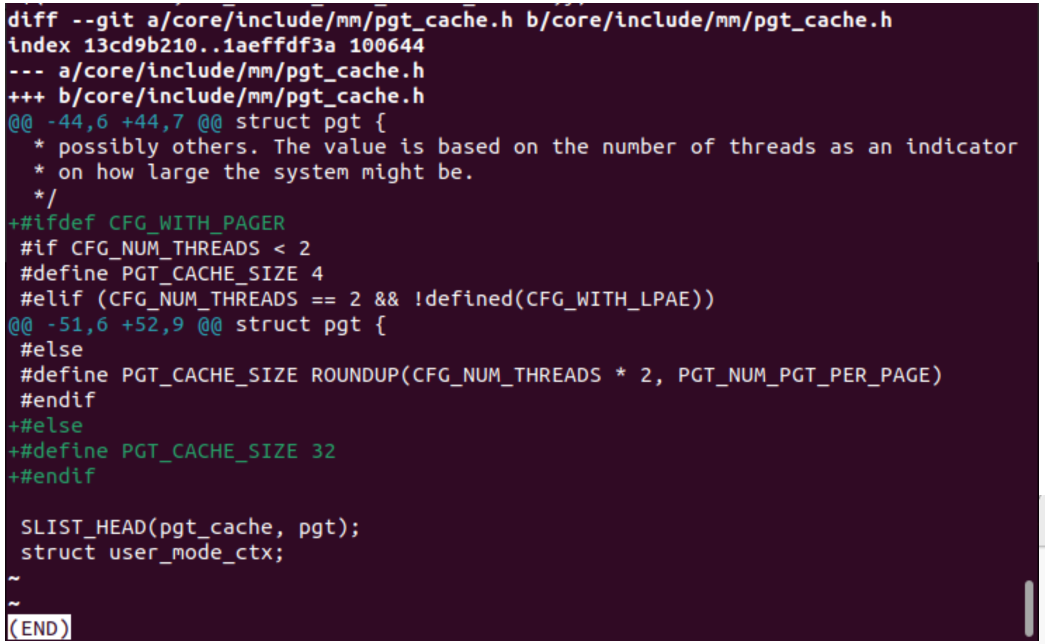
- https://github.com/mofanv/darknetz

1. Changes (continues on next 2 slides)

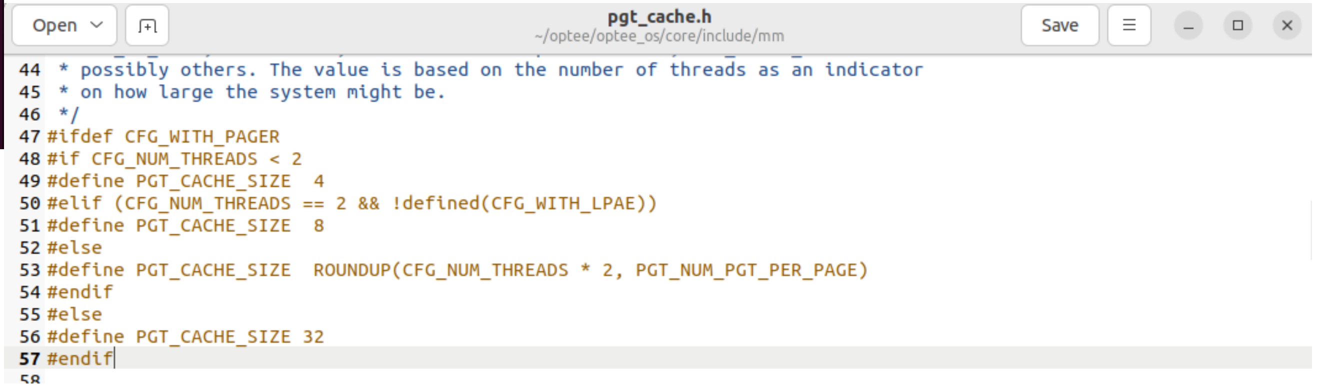
- https://optee.readthedocs.io/en/latest/building/prerequisites.html#prerequisites
- Stop before running the last 2 make commands
- Complete the changes on the following 2 slides

Installation Changes Cont.

- In $Optee_Path$/optee_os/core/include/mm/pgt.cache.h
  - Include the changes in green below



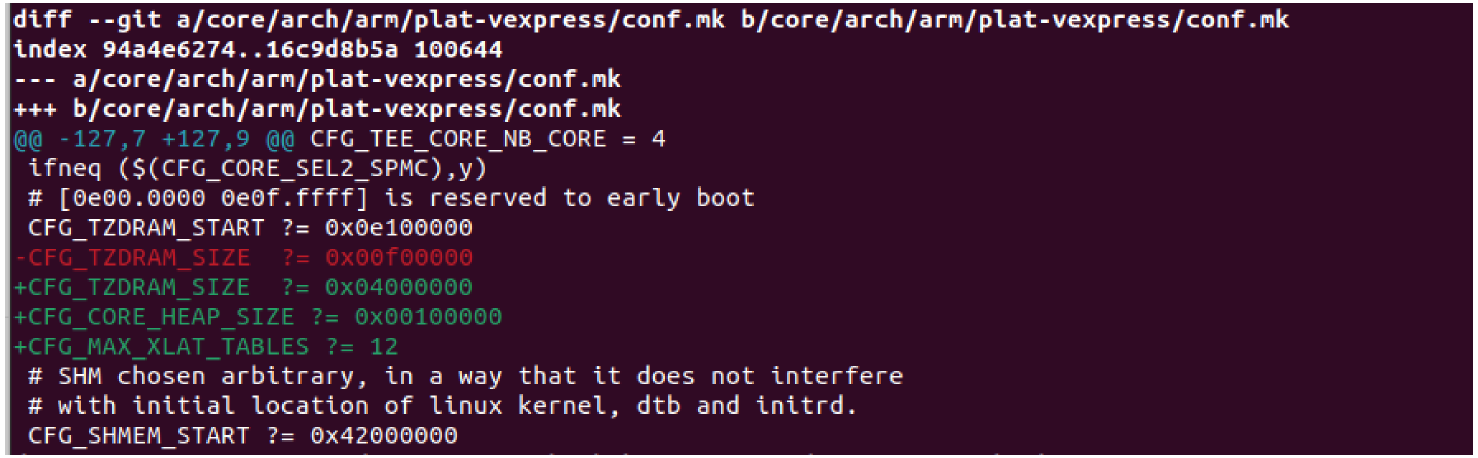
```
diff --git a/core/include/mm/pgt_cache.h b/core/include/mm/pgt_cache.h
index 13cd9b210..1aeffdf3a 100644
--- a/core/include/mm/pgt_cache.h
+++ b/core/include/mm/pgt_cache.h
@@ -44,6 +44,7 @@ struct pgt {
  * possibly others. The value is based on the number of threads as an indicator
  * on how large the system might be.
  */
+#ifdef CFG_WITH_PAGER
 #if CFG_NUM_THREADS < 2
 #define PGT_CACHE_SIZE 4
 #elif (CFG_NUM_THREADS == 2 && !defined(CFG_WITH_LPAE))
@@ -51,6 +52,9 @@ struct pgt {
 #else
 #define PGT_CACHE_SIZE ROUNDUP(CFG_NUM_THREADS * 2, PGT_NUM_PGT_PER_PAGE)
 #endif
+#else
+#define PGT_CACHE_SIZE 32
+#endif

 SLIST_HEAD(pgt_cache, pgt);
 struct user_mode_ctx;
~
~
(END)
```



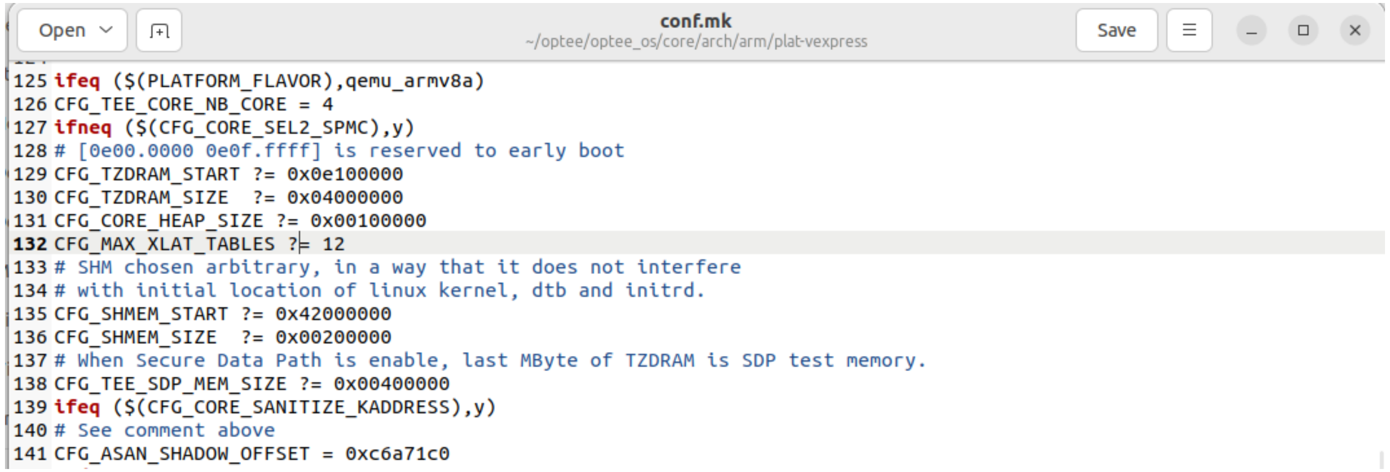
```
 * possibly others. The value is based on the number of threads as an indicator
 * on how large the system might be.
 */
#ifdef CFG_WITH_PAGER
#if CFG_NUM_THREADS < 2
#define PGT_CACHE_SIZE  4
#elif (CFG_NUM_THREADS == 2 && !defined(CFG_WITH_LPAE))
#define PGT_CACHE_SIZE  8
#else
#define PGT_CACHE_SIZE  ROUNDUP(CFG_NUM_THREADS * 2, PGT_NUM_PGT_PER_PAGE)
#endif
#else
#define PGT_CACHE_SIZE 32
#endif
```

INSTALLATION CHANGES CONT.

- In $Optee_Path$/optee_os/core/arch/arm/plat-vexpress/conf.mk
  - Remove the changes highlighted in red and add the changes highlighted in green below



```
diff --git a/core/arch/arm/plat-vexpress/conf.mk b/core/arch/arm/plat-vexpress/conf.mk
index 94a4e6274..16c9d8b5a 100644
--- a/core/arch/arm/plat-vexpress/conf.mk
+++ b/core/arch/arm/plat-vexpress/conf.mk
@@ -127,7 +127,9 @@ CFG_TEE_CORE_NB_CORE = 4
 ifneq ($(CFG_CORE_SEL2_SPMC),y)
 # [0e00.0000 0e0f.ffff] is reserved to early boot
 CFG_TZDRAM_START ?= 0x0e100000
-CFG_TZDRAM_SIZE  ?= 0x00f00000
+CFG_TZDRAM_SIZE  ?= 0x04000000
+CFG_CORE_HEAP_SIZE ?= 0x00100000
+CFG_MAX_XLAT_TABLES ?= 12
 # SHM chosen arbitrary, in a way that it does not interfere
 # with initial location of linux kernel, dtb and initrd.
 CFG_SHMEM_START ?= 0x42000000
```



```
ifeq ($(PLATFORM_FLAVOR),qemu_armv8a)
CFG_TEE_CORE_NB_CORE = 4
ifneq ($(CFG_CORE_SEL2_SPMC),y)
# [0e00.0000 0e0f.ffff] is reserved to early boot
CFG_TZDRAM_START ?= 0x0e100000
CFG_TZDRAM_SIZE  ?= 0x04000000
CFG_CORE_HEAP_SIZE ?= 0x00100000
CFG_MAX_XLAT_TABLES ?= 12
# SHM chosen arbitrary, in a way that it does not interfere
# with initial location of linux kernel, dtb and initrd.
CFG_SHMEM_START ?= 0x42000000
CFG_SHMEM_SIZE  ?= 0x00200000
# When Secure Data Path is enable, last MByte of TZDRAM is SDP test memory.
CFG_TEE_SDP_MEM_SIZE ?= 0x00400000
ifeq ($(CFG_CORE_SANITIZE_KADDRESS),y)
# See comment above
CFG_ASAN_SHADOW_OFFSET = 0xc6a71c0
```

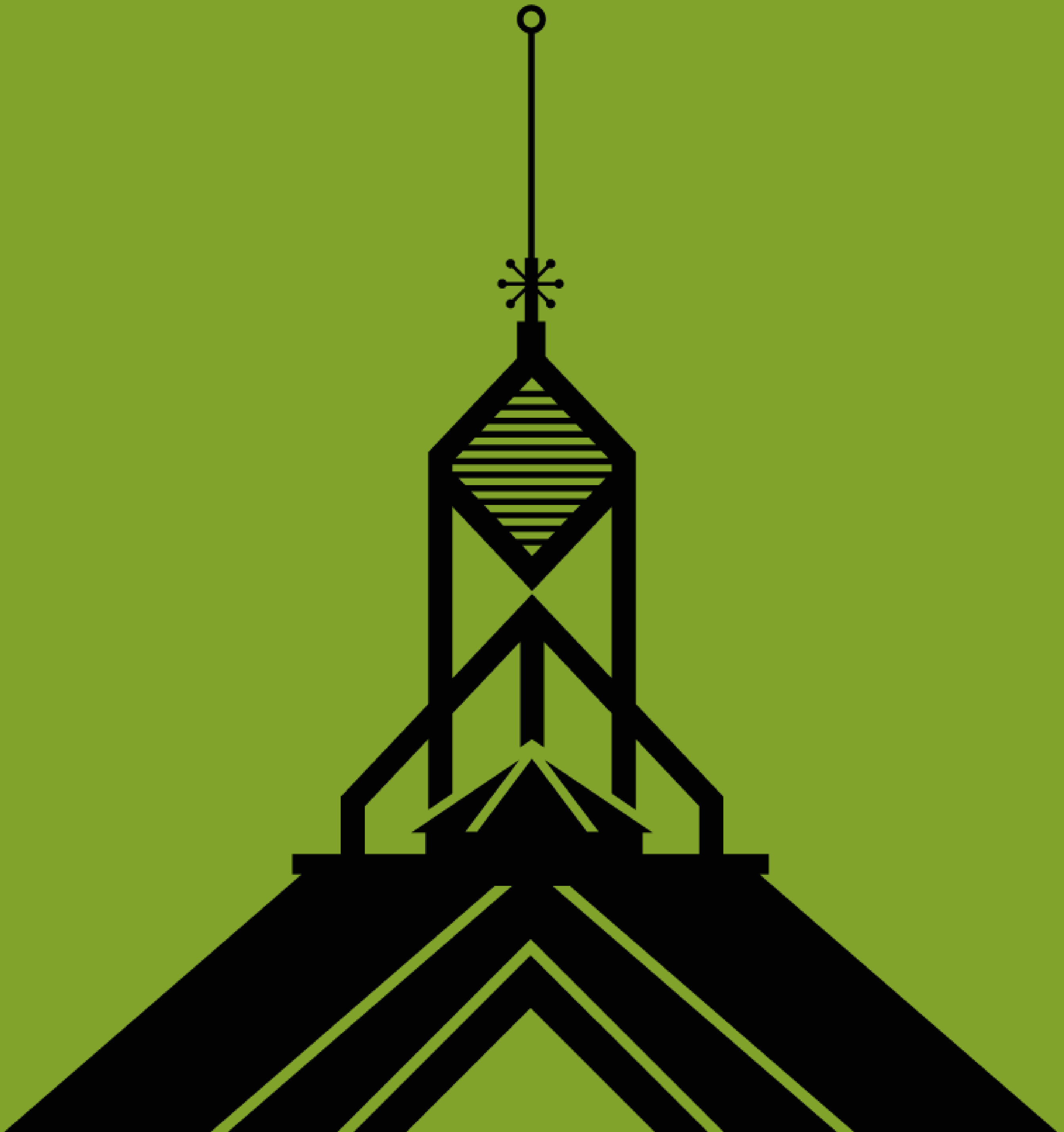
Installation Changes Cont.

Continue with the prerequisites page and run the two ‘make’ commands at the end.

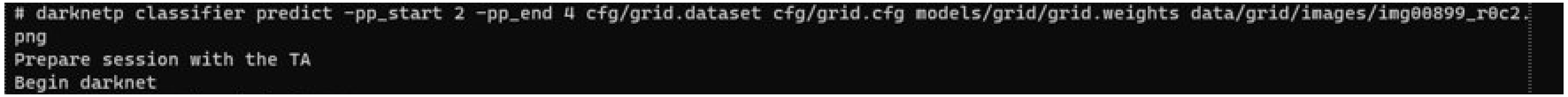
Then skip to step 8 in https://optee.readthedocs.io/en/latest/building/gits/build.html#get-and-build-the-solution and step 9 after completing step 8.

After, return to the main DarkneTZ page, https://github.com/mofanv/darknetz and continue from (2) Build DarkneTZ

WEEK 10 – DARKNETZ FUNCTIONS

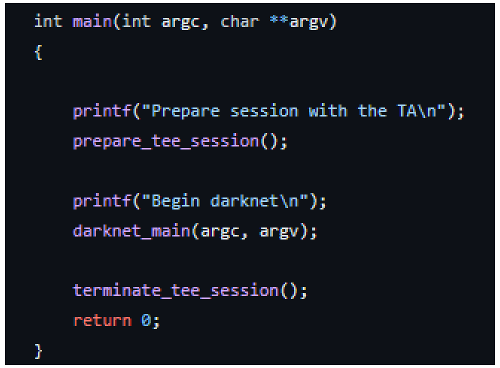


BASIC CONTROL FLOW OF DARKNETZ FOR GRID EXAMPLE

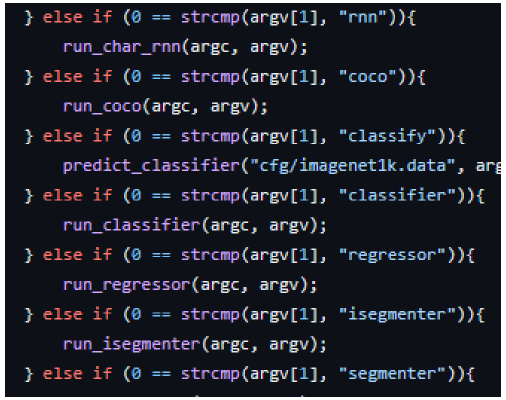


```
# darknetp classifier predict -pp_start 2 -pp_end 4 cfg/grid.dataset cfg/grid.cfg models/grid/grid.weights data/grid/images/img00899_r0c2.
png
Prepare session with the TA
Begin darknet
```

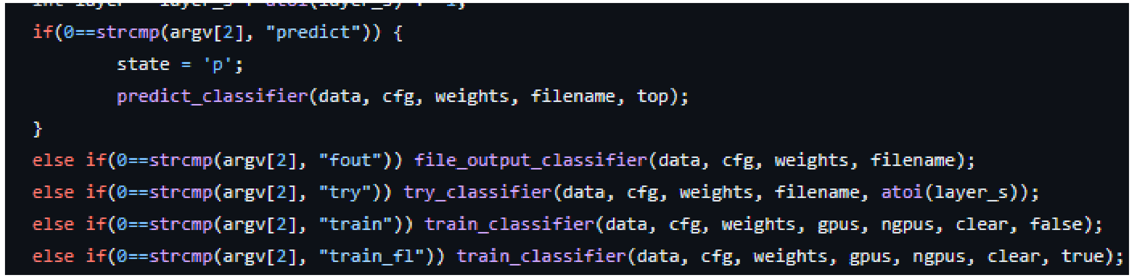
- Darknetp main function
  - Prepares the TEE session with prepare_tee_session()
  - Calls darknet_main()
    - Checks first command line argument to check what example to run and then calls example
- run_classifier (examples/classifier.c)
  - Scans in command line arguments for partition points, weights, filename, data, etc.
  - Calls corresponding classifier function based on second command line argument (‘predict’ in our case, so it calls ‘predict_classifier’)



```
int main(int argc, char **argv)
{

    printf("Prepare session with the TA\n");
    prepare_tee_session();

    printf("Begin darknet\n");
    darknet_main(argc, argv);

    terminate_tee_session();
    return 0;
}
```



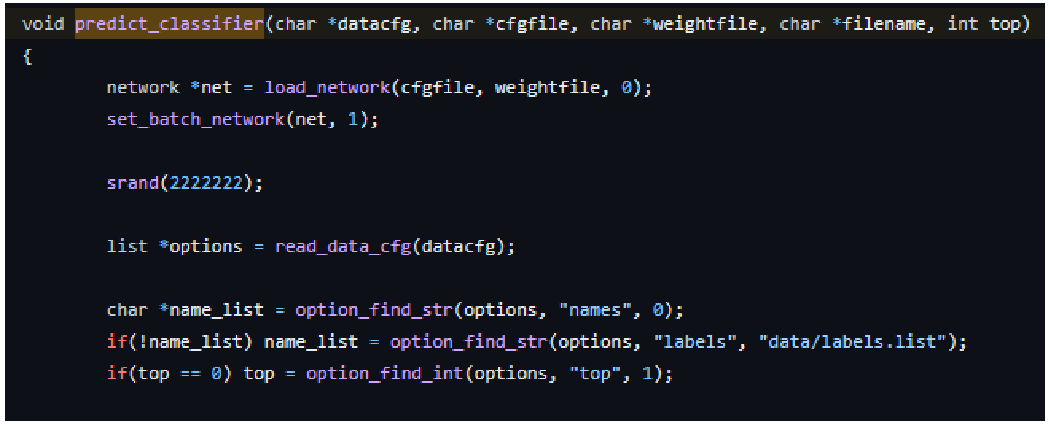
```
} else if (0 == strcmp(argv[1], "rnn")){
    run_char_rnn(argc, argv);
} else if (0 == strcmp(argv[1], "coco")){
    run_coco(argc, argv);
} else if (0 == strcmp(argv[1], "classify")){
    predict_classifier("cfg/imagenet1k.data", arg
} else if (0 == strcmp(argv[1], "classifier")){
    run_classifier(argc, argv);
} else if (0 == strcmp(argv[1], "regressor")){
    run_regressor(argc, argv);
} else if (0 == strcmp(argv[1], "isegmenter")){
    run_isegmenter(argc, argv);
} else if (0 == strcmp(argv[1], "segmenter")){
```



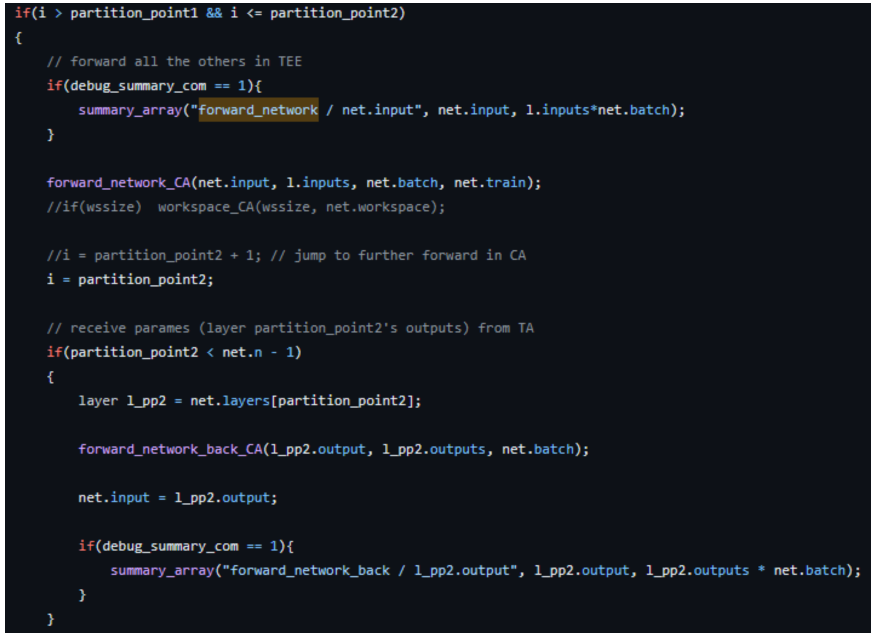
```
if(0==strcmp(argv[2], "predict")) {
        state = 'p';
        predict_classifier(data, cfg, weights, filename, top);
}
else if(0==strcmp(argv[2], "fout")) file_output_classifier(data, cfg, weights, filename);
else if(0==strcmp(argv[2], "try")) try_classifier(data, cfg, weights, filename, atoi(layer_s));
else if(0==strcmp(argv[2], "train")) train_classifier(data, cfg, weights, gpus, ngpus, clear, false);
else if(0==strcmp(argv[2], "train_fl")) train_classifier(data, cfg, weights, gpus, ngpus, clear, true);
```

BASIC CONTROL FLOW OF DARKNETZ FOR GRID EXAMPLE

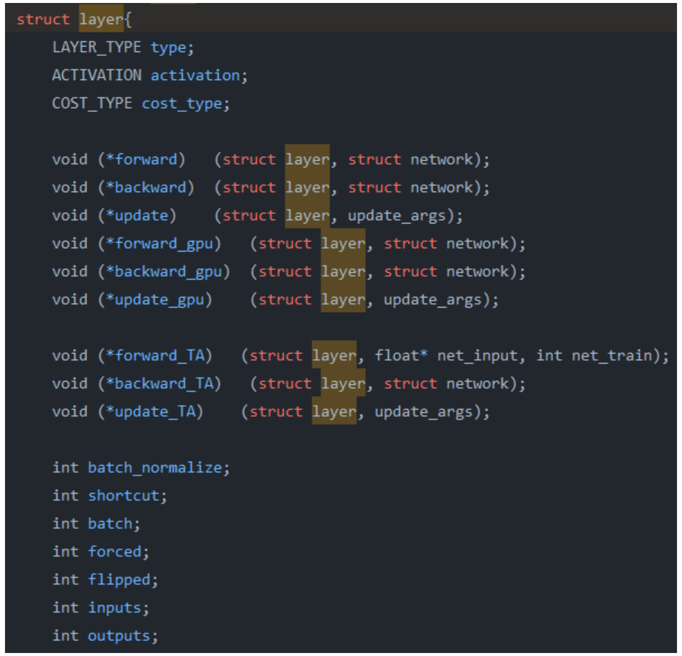
- predict_classifier
    - Loads the network from cfg file and weights file with load_network()
        - Creates network struct that contains network information
            - Contains array of layers containing individual layer information
    - Loads weights
    - Gets filename to run inference on
    - Calls network_predict(), with network struct as argument
        - Calls forward_network() which runs the network with the appropriate layers in the REE and TEE and returns the results



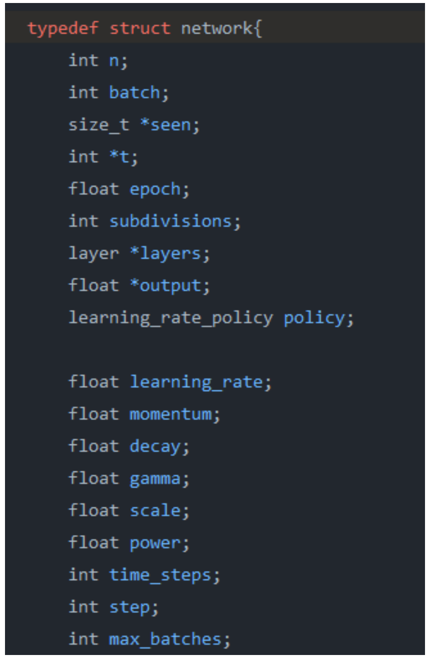
```
void predict_classifier(char *datacfg, char *cfgfile, char *weightfile, char *filename, int top)
{
        network *net = load_network(cfgfile, weightfile, 0);
        set_batch_network(net, 1);

        srand(2222222);

        list *options = read_data_cfg(datacfg);

        char *name_list = option_find_str(options, "names", 0);
        if(!name_list) name_list = option_find_str(options, "labels", "data/labels.list");
        if(top == 0) top = option_find_int(options, "top", 1);
```



```
if(i > partition_point1 && i <= partition_point2)
{
    // forward all the others in TEE
    if(debug_summary_com == 1){
        summary_array("forward_network / net.input", net.input, l.inputs*net.batch);
    }

    forward_network_CA(net.input, l.inputs, net.batch, net.train);
    //if(wssize)  workspace_CA(wssize, net.workspace);

    //i = partition_point2 + 1; // jump to further forward in CA
    i = partition_point2;

    // receive parames (layer partition_point2's outputs) from TA
    if(partition_point2 < net.n - 1)
    {
        layer l_pp2 = net.layers[partition_point2];

        forward_network_back_CA(l_pp2.output, l_pp2.outputs, net.batch);

        net.input = l_pp2.output;

        if(debug_summary_com == 1){
            summary_array("forward_network_back / l_pp2.output", l_pp2.output, l_pp2.outputs * net.batch);
        }
    }
```



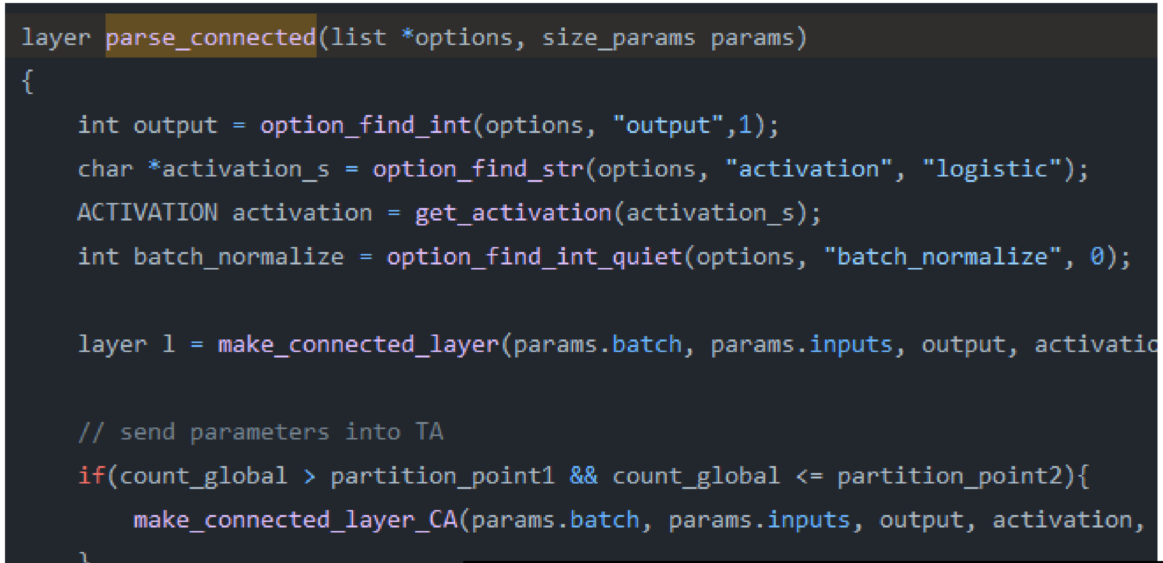
```
struct layer{
    LAYER_TYPE type;
    ACTIVATION activation;
    COST_TYPE cost_type;

    void (*forward)   (struct layer, struct network);
    void (*backward)  (struct layer, struct network);
    void (*update)    (struct layer, update_args);
    void (*forward_gpu)   (struct layer, struct network);
    void (*backward_gpu)  (struct layer, struct network);
    void (*update_gpu)    (struct layer, update_args);

    void (*forward_TA)   (struct layer, float* net_input, int net_train);
    void (*backward_TA)   (struct layer, struct network);
    void (*update_TA)    (struct layer, update_args);

    int batch_normalize;
    int shortcut;
    int batch;
    int forced;
    int flipped;
    int inputs;
    int outputs;
```



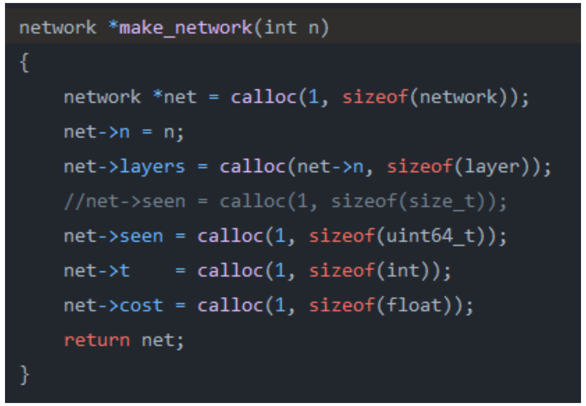
```
typedef struct network{
    int n;
    int batch;
    size_t *seen;
    int *t;
    float epoch;
    int subdivisions;
    layer *layers;
    float *output;
    learning_rate_policy policy;

    float learning_rate;
    float momentum;
    float decay;
    float gamma;
    float scale;
    float power;
    int time_steps;
    int step;
    int max_batches;
```

DARKNETZ LAYER FUNCTIONS

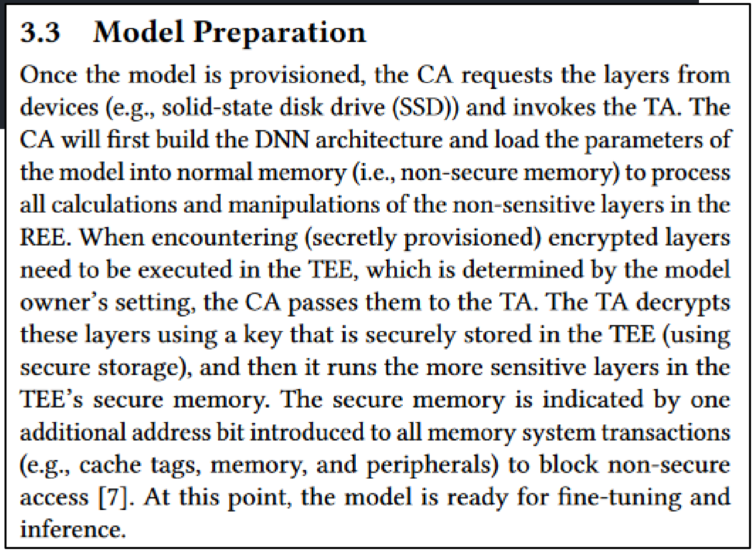
- make_x_layer and make_x_layer_CA
  - These are used by parser.c to generate a network from a config file
    - Parser.c uses a global variable to keep track of partition points
- Layers inserted into a network struct, along with other values
- Layer Types:
  - Convolutional
  - Deconvolutional
  - Local
  - Activation
  - Logistic
  - L2norm
  - Rnn
  - Gru
  - Lstm



```
layer parse_connected(list *options, size_params params)
{
    int output = option_find_int(options, "output",1);
    char *activation_s = option_find_str(options, "activation", "logistic");
    ACTIVATION activation = get_activation(activation_s);
    int batch_normalize = option_find_int_quiet(options, "batch_normalize", 0);

    layer l = make_connected_layer(params.batch, params.inputs, output, activatio

    // send parameters into TA
    if(count_global > partition_point1 && count_global <= partition_point2){
        make_connected_layer_CA(params.batch, params.inputs, output, activation,
    }
    return l;
}
```



```
network *make_network(int n)
{
    network *net = calloc(1, sizeof(network));
    net->n = n;
    net->layers = calloc(net->n, sizeof(layer));
    //net->seen = calloc(1, sizeof(size_t));
    net->seen = calloc(1, sizeof(uint64_t));
    net->t    = calloc(1, sizeof(int));
    net->cost = calloc(1, sizeof(float));
    return net;
}
```



3.3 Model Preparation

Once the model is provisioned, the CA requests the layers from devices (e.g., solid-state disk drive (SSD)) and invokes the TA. The CA will first build the DNN architecture and load the parameters of the model into normal memory (i.e., non-secure memory) to process all calculations and manipulations of the non-sensitive layers in the REE. When encountering (secretly provisioned) encrypted layers need to be executed in the TEE, which is determined by the model owner's setting, the CA passes them to the TA. The TA decrypts these layers using a key that is securely stored in the TEE (using secure storage), and then it runs the more sensitive layers in the TEE's secure memory. The secure memory is indicated by one additional address bit introduced to all memory system transactions (e.g., cache tags, memory, and peripherals) to block non-secure access [7]. At this point, the model is ready for fine-tuning and inference.