Introduction to Program 3

ECS 170

Hannah Brown

Nodes Expanded: 16,662
Time Taken: 12,4542

Bidirectional A*

- Two searches: one starting from the initial state and one from the goal state

- Requires more bookkeeping: an open and closed list for each search, and cost has to be calculated carefully for both g(n) and h(n)

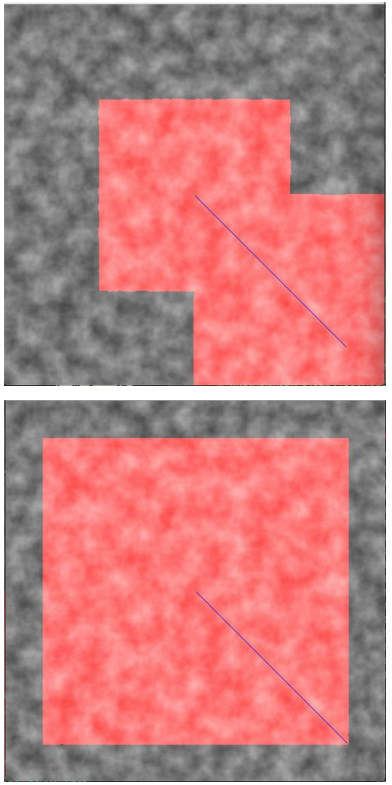
    - Priorityqueues used for open lists and hashmaps used for closed lists to minimize bookkeeping cost

- $O(b^{d/2})$ instead of $O(b^d)$

- My heuristic: h(k) = (k*current height)/256,

    - k = Chebyshev distance from current position to goal

    - Admissible but not consistent, so there were some node re-expansions



Bidirectional Dijkstra vs. Normal Dijkstra

When to Stop Looking

- When the two frontiers intersect, the shortest path is guaranteed to have been found

- The position where the two frontiers intersect is ***not guaranteed*** to be on that shortest path!

- Keep track of the min cost path seen so far and the nodes in that path

- Terminate when: min_cost ≤ max(min(f_start(x)), min(f_goal(x)))

    - x is a node in both frontiers

Termination condition source: https://arxiv.org/pdf/cs/9712102.pdf

Bidirectional A* on Mt. St. Helens

Matthew Marlow

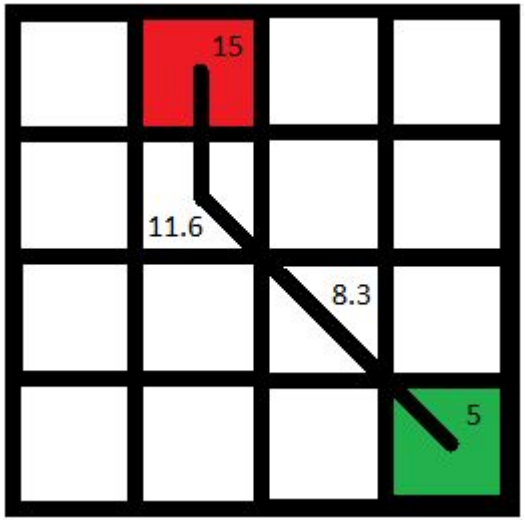
Nodes Expanded: 455
Time Taken: 78,938

Heuristic Function

```
// return 0 if we're comparing the goal state to the goal state
if (pt1.x == pt2.x && pt1.y == pt2.y)
            return 0;

// otherwise calculate the heuristic
// h = avg step cost * min number of steps to goal.
double startHeight = map.getTile(pt1);
double endHeight = map.getTile(pt2);
double delta = endHeight - startHeight;
int chebDist = Math.max(Math.abs(pt2.x - pt1.x), Math.abs(pt2.y - pt1.y));
double avgStepHeight = delta / chebDist;
double h = 0.0;
for (int x = 0; x < chebDist; x++) {
    h += startHeight / (startHeight + avgStepHeight + 1);
    startHeight += avgStepHeight;
}

return h * 0.95;
```



Increasing Efficiency With Bi-Directional Graph Search

Fringe - TreeMap<Double, LinkedList<Point>> - automatically inserts in key-order, one for each direction

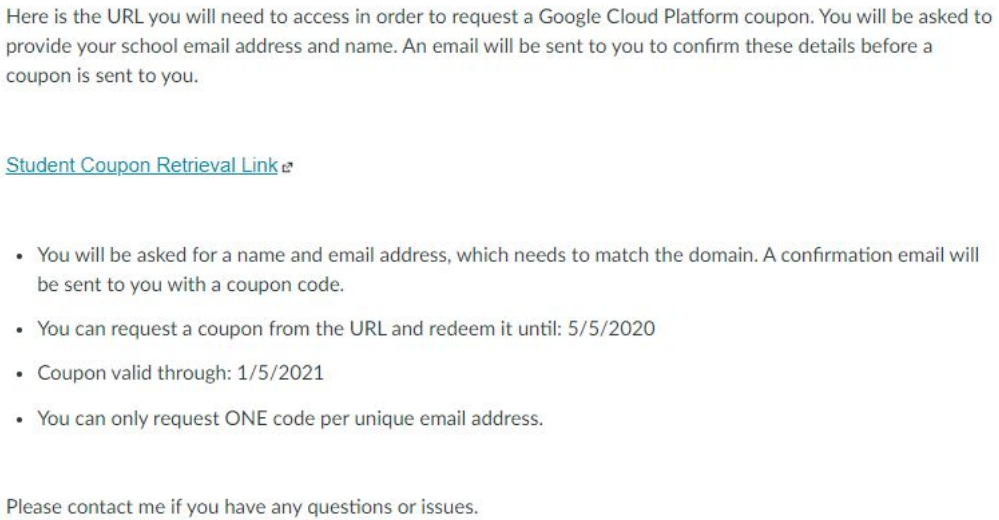
Closed List - Boolean[][] - constant lookup/write time, one for each direction. Stops searching when the first tile in one direction’s fringe is in the closed list of the other direction.

G Value Table - Double[][] - Like with the closed list, using a 2d array is more space complex, but less time complex. Only 1 required for Bi-Directional search since the two directions will never overlap

Introduction to Google Cloud

0. Redeem Credits

- Link in announcements
- Credits are emailed and get redeemed to **default** Google account
- **Only redeem credits once, there is only enough for 1 coupon per student**
- Worst case there is free trial credits



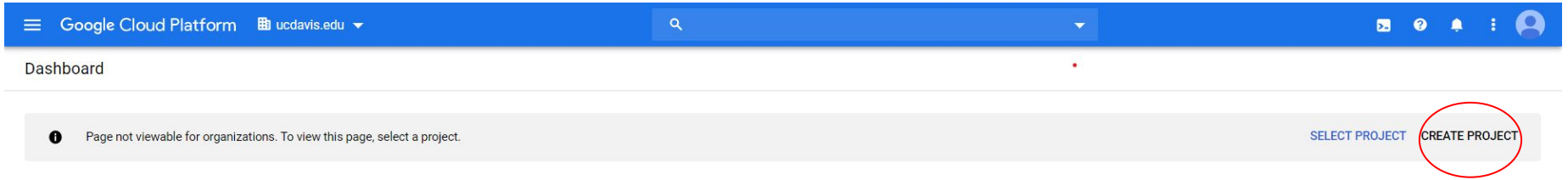
Here is the URL you will need to access in order to request a Google Cloud Platform coupon. You will be asked to provide your school email address and name. An email will be sent to you to confirm these details before a coupon is sent to you.

Student Coupon Retrieval Link

- You will be asked for a name and email address, which needs to match the domain. A confirmation email will be sent to you with a coupon code.
- You can request a coupon from the URL and redeem it until: 5/5/2020
- Coupon valid through: 1/5/2021
- You can only request ONE code per unique email address.

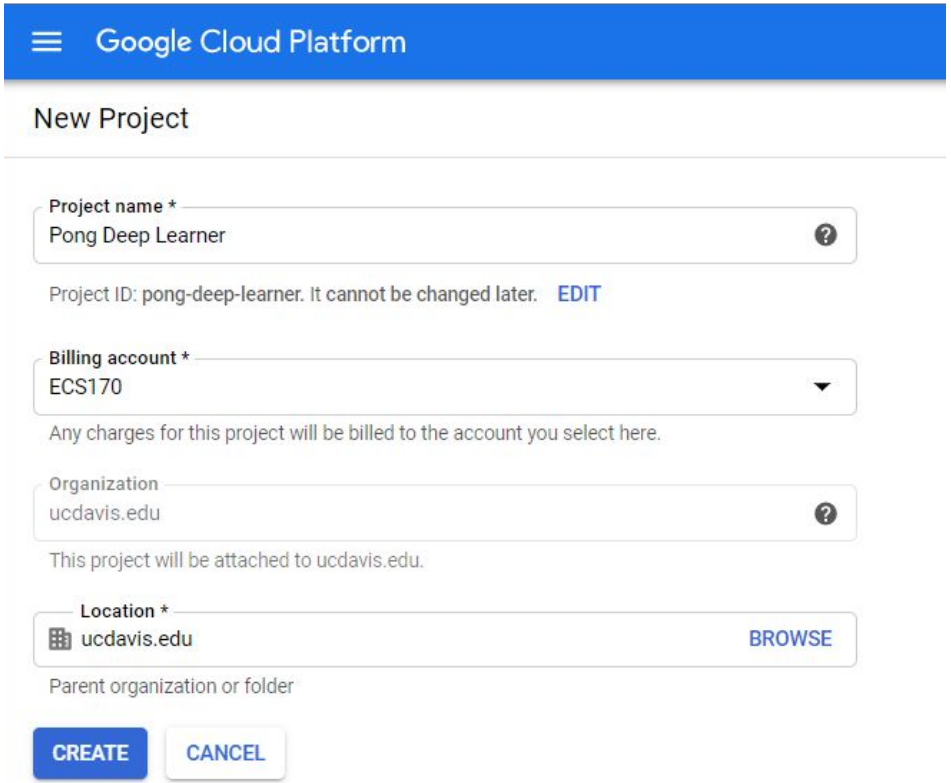
Please contact me if you have any questions or issues.

1. Create a Project



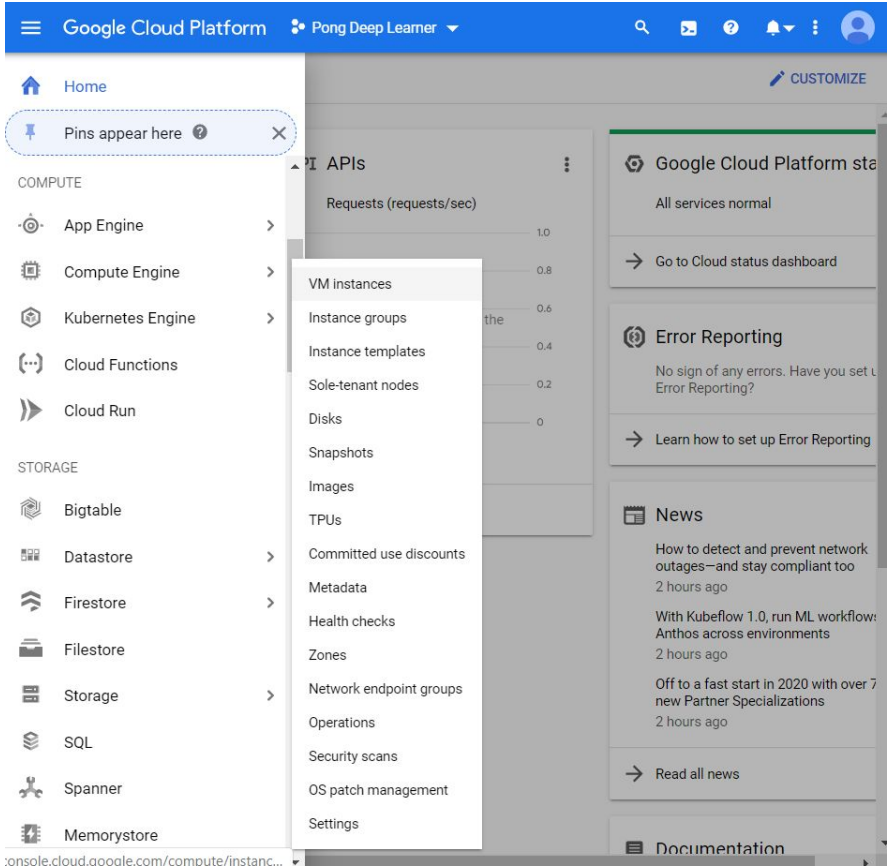
2. Project Settings

- Set your new project to match the right
- **Make sure to set billing account to ECS170**
  - **Go through credits redemption first**

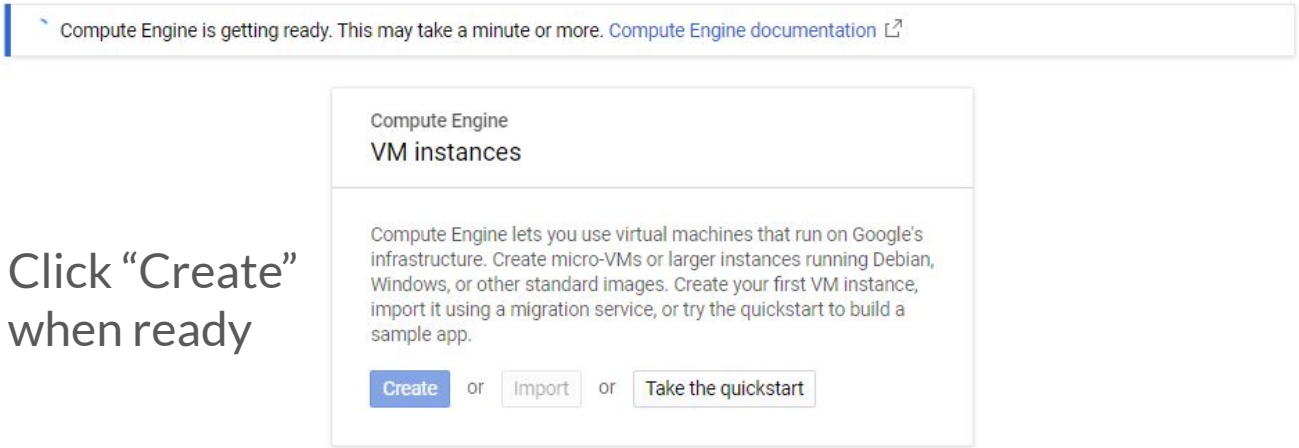


3. Make a VM

- Make a virtual machine
- Hamburger icon
  - Compute Engine
    - VM Instances



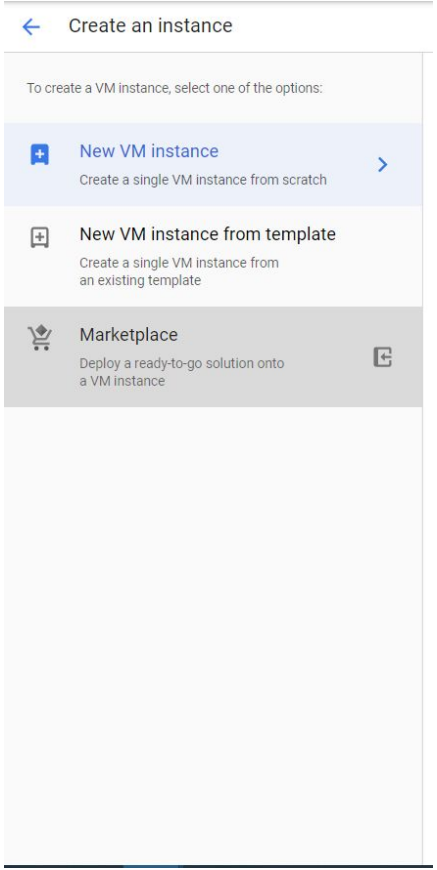
4. Wait for Compute Engine to Initialize



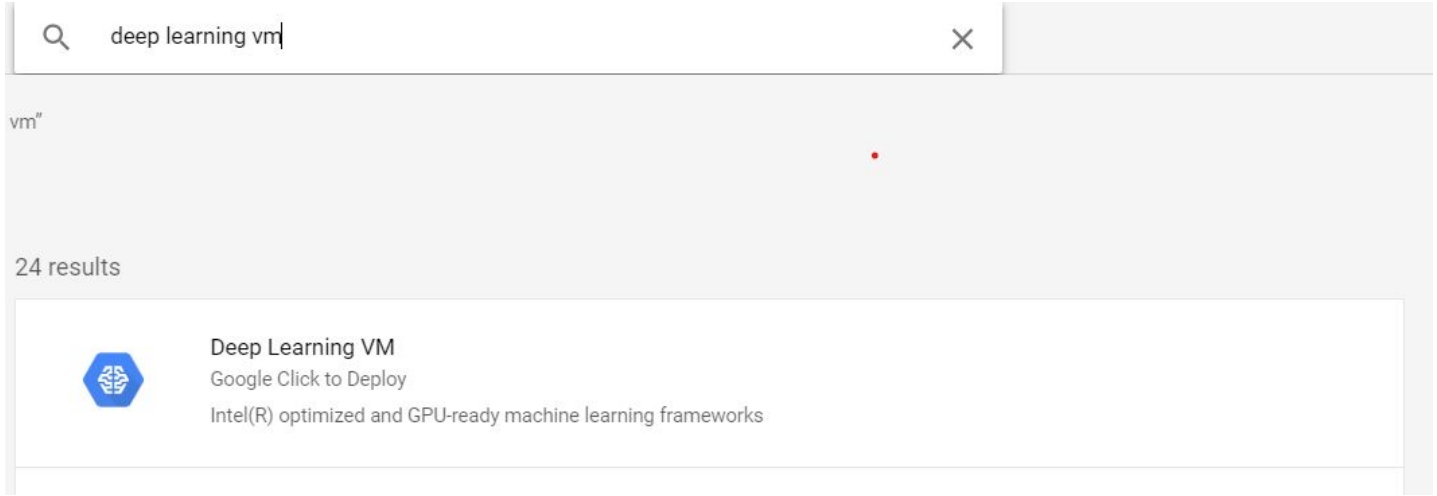
Click “Create” when ready

5. Making VM

- Click “Marketplace” to get a VM from the marketplace

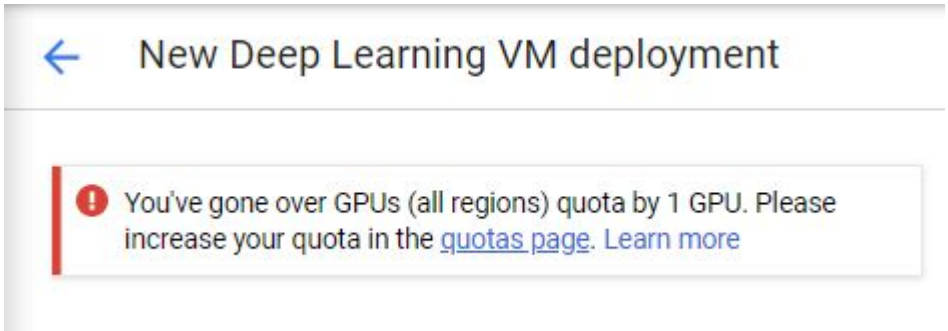


6. Use “Deep Learning VM” from Google

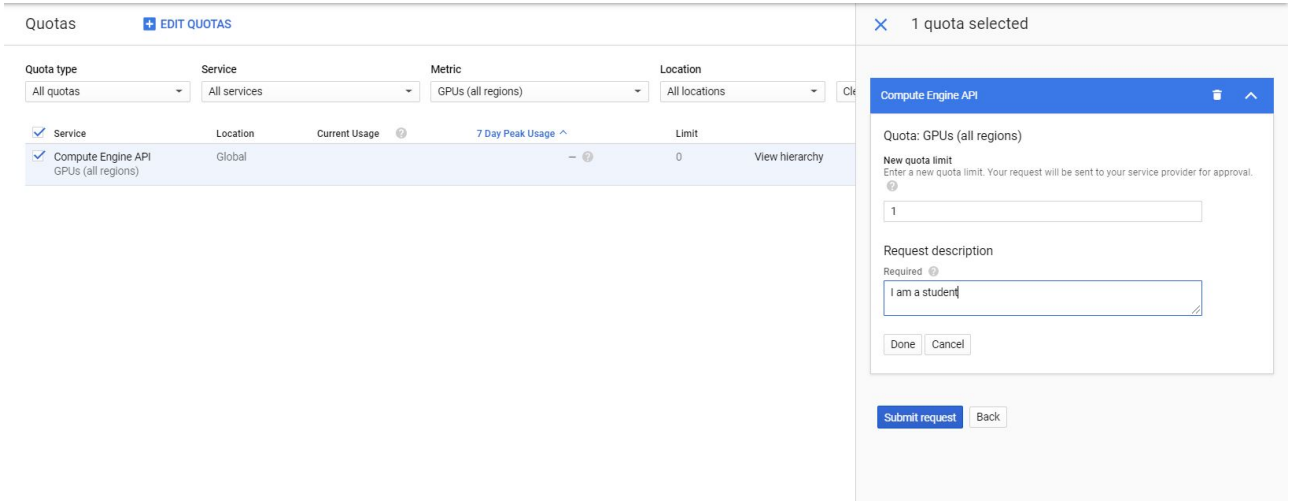


7. GPU Quota

- If you encounter the warning on the right click on “quotas page”
- Neural networks train much faster on GPU’s



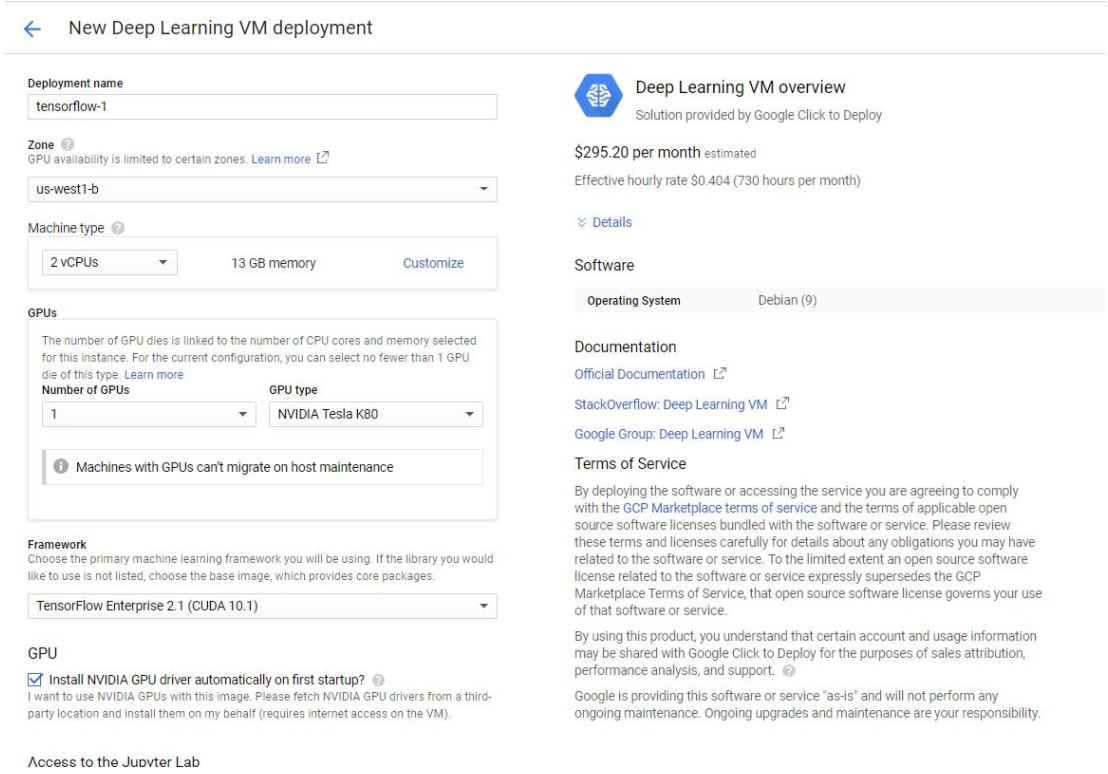
8. GPU Quota Request



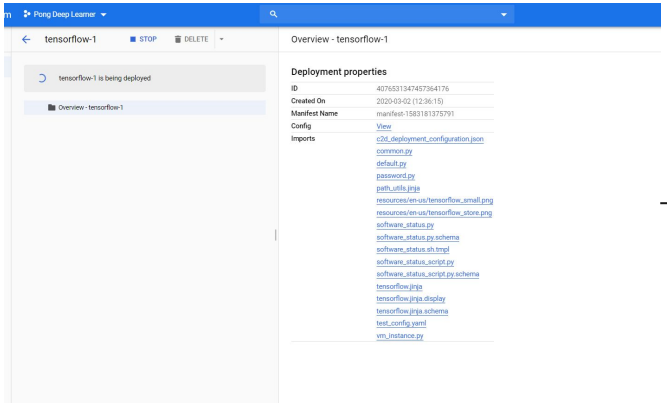
- Request an increase in GPUs (all regions) to 1 GPU
- This takes roughly 10 minutes for Google to accept (may take longer)

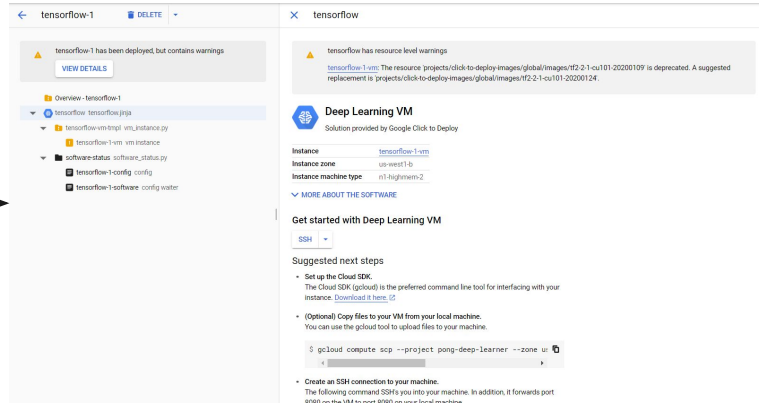
9. VM Settings

- Make your VM match the right after GPU quota increased
- Important
  - GPUs: 1
  - Framework: Tensorflow 2.1
- (This is the page from step 7)

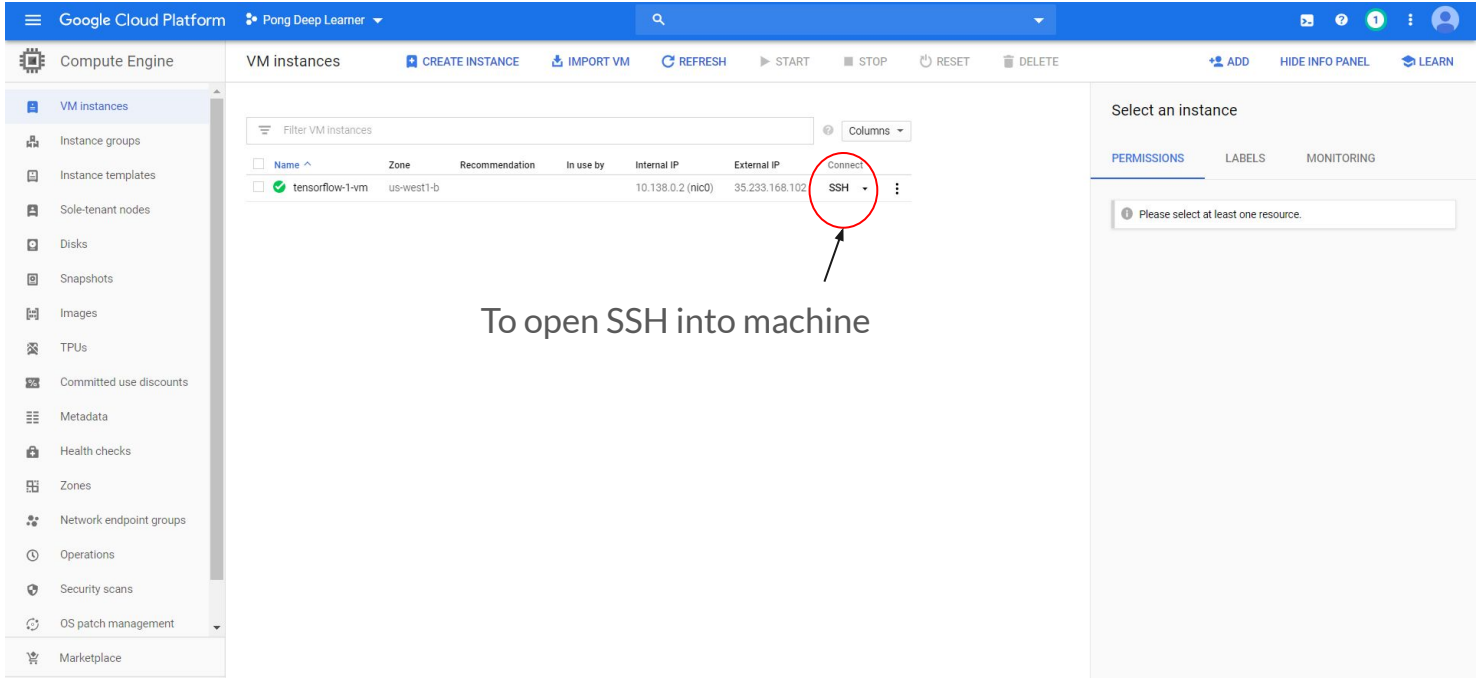


10. Await Deployment

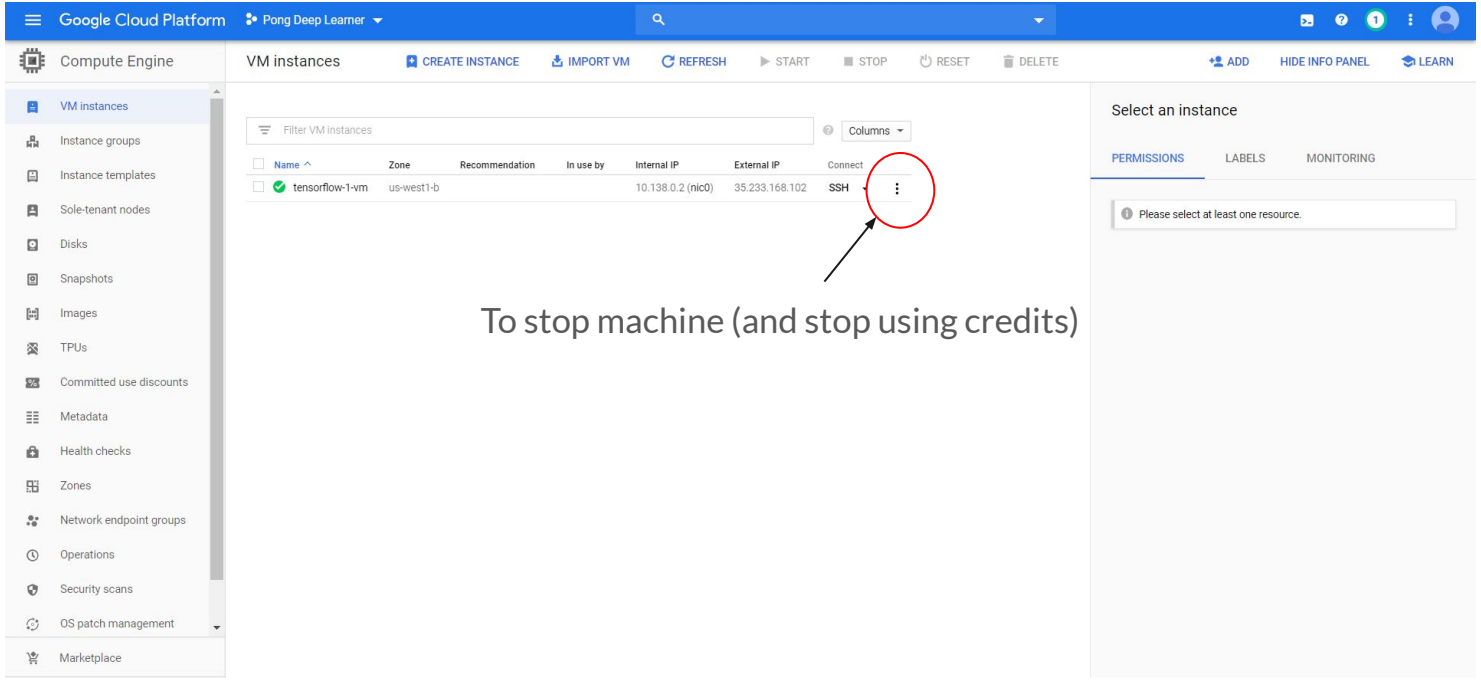




11. SSH into VM



11. SSH into VM



Loading the Homework

- For ease of use, the homework is duplicated on GitHub:
- https://github.com/kurt-schneider/ECS-170-Program-3-Starter-Code
  - git clone 'https://github.com/kurt-schneider/ECS-170-Program-3-Starter-Code'
- Make sure to do the additional installs listed in the pdf
- The code will not run until you have implemented all the functions required
- You may have to take additional steps to use the "-g" option for test_dqn_pong.py. The gui is not required

Miscellanea

- Use GitHub
- Watch your model train for a bit first to make sure its improving
- Train model using ‘nohup python3 run_dqn_pong.py’
  - Do **not** stop the machine if you want nohup to keep running
- ‘ALE_del’ error/warning on termination can be ignored
  - Something in the library seems to have trouble deleting itself

Questions

PyTorch Crash Course

CUDA

- CUDA is how pytorch interacts with the GPU
- GPUs are (generally) better for neural network training

CUDA

- CUDA is how pytorch interacts with the GPU
- GPUs are (generally) better for neural network training
    - **Why?**

CUDA

- CUDA is how pytorch interacts with the GPU
- GPUs are (generally) better for neural network training
  - Why?
  - GPUs optimized for matrix calculations, parallel processing, high data bandwidth (exactly what we need for neural nets)
- `USE_CUDA = torch.cuda.is_available()`

Tensors

- Similar to (numpy) arrays
- Hold pieces of data (e.g. images, network outputs)
- `action = torch.LongTensor(action)`
- `action.size()`
- `action.view(-1, 10)`
- `action.detach().cpu().numpy()`
- torch.Tensor() vs torch.cuda.Tensor() vs torch.Tensor().cuda() ?

Neural Networks

- Models (neural networks) can also be sent to the GPU using .cuda() !

```
class Neural_Network(nn.Module):
    def __init__(self, ):
        super(Neural_Network, self).__init__()
        # parameters
        # TODO: parameters can be parameterized instead of declaring them here
        self.inputSize = 2
        self.outputSize = 1
        self.hiddenSize = 3

        # weights
        self.W1 = torch.randn(self.inputSize, self.hiddenSize) # 2 X 3 tensor
        self.W2 = torch.randn(self.hiddenSize, self.outputSize) # 3 X 1 tensor

    def forward(self, X):
        self.z = torch.matmul(X, self.W1)
        self.z2 = self.sigmoid(self.z) # activation function
        self.z3 = torch.matmul(self.z2, self.W2)
        o = self.sigmoid(self.z3) # final activation function
        return o
```

Training Neural Networks

```
loss = compute_td_loss(model, target_model, batch_size,
                       gamma, replay_buffer) # gets loss
optimizer.zero_grad() # resets the optimizer gradient
loss.backward() # backpropagates the loss through the net (i.e. update)
optimizer.step() # updates optimizer parameters
```

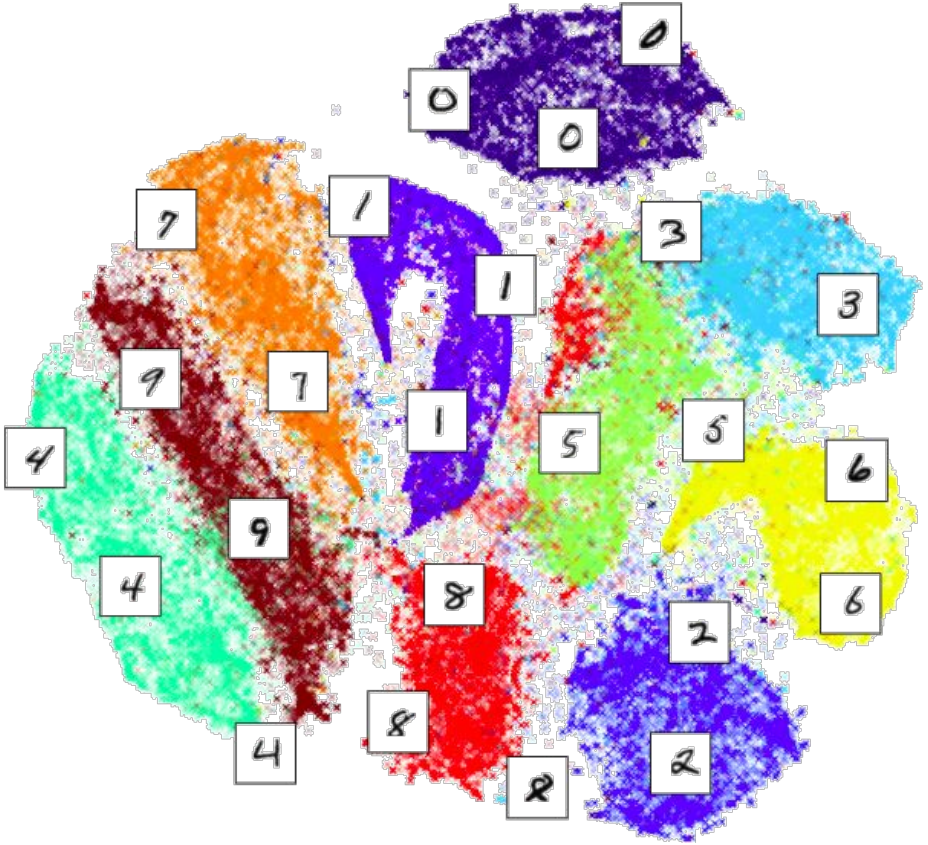
Questions

Important Notes

- Training a model takes a while and this is **unavoidable**
  - **Start early!**
  - No trained model => no good grade 😢
- Watch results for a few ten thousand frames
  - No decrease => you probably have a mistake
- Reward **may not** strictly increase, loss **may not** strictly decrease
  - Moving target of minimizing model loss + maximizing reward
  - Target network helps stabilize this
- Try adjusting the parameters
  - E.g. learning rate, epsilon, gamma, etc

Bonus Points

- XAI is an active research area
- Explaining “black box” models is difficult
- Try to use your human knowledge of Pong and observation of the model to explain what it may have learned
    - Give evidence for it with the learned embedding



TSNE Visualization for MNIST Embedding

Good Luck!