Information Theory Recitation

Siming He, ESE 5460

November 2023

Contents

1 Elements of Information Theory

Information theory was developed as a mathematical theory of communication. A communication system can be divided into five parts:

source $\rightarrow$ **transmitter** $\rightarrow$ **channel** $\rightarrow$ **receiver** $\rightarrow$ **destination**

1. **Source** is the message that we want to send to the destination, e.g. text messages, images, videos

2. **Transmitter** converts the message into some signals that can be transmitted. For example, Morse code converts numbers and English alphabets into dots and dashes. Dots and dashes signals can be easily transmitted by short-duration light/sound and long-duration light/sound.

3. **Channel** is the medium used to transmit signal. For example, air transmits light/sound. And wires transmits pulses of voltages.

4. **Receiver** performs an inverse operation of transmitter to reconstruct messages from signals.

5. **Destination** would be the person who receives and reads your text.

Two fundamental questions that are essential to communication system and can be answered by information theory.

1. **What is the maximal compression of data?** We want the communication system to be energy-efficient, i.e., spend the minimum amount of energy to transmit messages. To achieve this, we want to (maximally and losslessly) **compress** our messages.

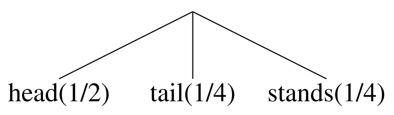
2. **What is the maximum rate that data can be transmitted?** The maximally compressed message would work well if the channel is noiseless. However, if the channel is noisy, compressed messages may be easily corrupted by the noise during transmission. To make the signal robust to noise, we also want to add **redundancy** into the signal. So the maximum rate of transmission depends on both compression and necessary redundancy.

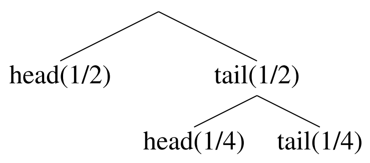
To understand the design of transmitter or encoder, we have to understand the information in the source or input. However, the concept of information is vague and broad and is hard to be formulated mathematically. Instead, we can define a measure of the amount of information in a random variable. For a probability distribution $P^{(n)} = (p_1, \cdots, p_n)$ over an alphabet of size n, we want to find an information measure $H_n(p_1, \cdots, p_n)$ characterising the information content of chance experiment with $P^{(n)}$. Shannon argued that there are three required properties of desirable information measure:

1. $H_n(p_1, \cdots, p_n)$ is a continuous function of $p_1, \cdots, p_n$. This is intuitive since a small change in the probability distribution shouldn't lead to a sudden jump in information content.

2. For $H_n(p_1, \cdots, p_n)$ where $p_1 = \cdots = p_n$, H_n should be a monotonic increasing function of n. Intuitively, more events leads to more uncertainty and more information. Rolling a dice seems to be more complex than flipping a coin.

3. The measure should have $H_n(p_1, \cdots, p_n) = H_{n-1}(p_1 + p_2, p_3, \cdots, p_n) + (p_1 + p_2)H_2(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2})$. For example, $H_3(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ should be equal to $H_2(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2}H_2(\frac{1}{2}, \frac{1}{2})$. Intuitively, if the information measure measures the amount of information in a chance outcome, how the source produces such outcome shouldn't change the amount of information. For example, the source can flip a magical coin that is head with probability $\frac{1}{2}$, is tail with probability $\frac{1}{4}$, stands up with probability $\frac{1}{4}$. The source can also flip a fair coin and flip the coin again if the first flip results in tail. Those two process gives the same chance outcome so should contain the same amount of information. For the second process, we firstly gain the information of $H_2(\frac{1}{2}, \frac{1}{2})$. Then with probability $\frac{1}{2}$, we gain additional information $H_2(\frac{1}{2}, \frac{1}{2})$ from the second round.





Definition 1.1 (Entropy)**.** The only H satisfying the three properties above is in the form of $H = -K\sum_{i=1}^{n} p_i \log p_i$. K is basically the unit of the measure and can be 1. Then, we have the definition of entropy:

$$H_n(p_1, \cdots, p_n) = -\sum_{i=1}^{n} p_i \log p_i$$

For a random variable X, we can also write the entropy as

$$H(X) = -\sum_{x} p(x) \log p(x)$$

Remark. The quantity $-\log p(x)$ can be viewed as the information content of event x. The quantity aligns with our intuition that rare events should contain more information $-\log p(x) > -\log p(y)$ when $p(x) < p(y)$. Entropy can be viewed as the expected information content from a random variable.

Remark. Another view of entropy is that entropy of a random variable is a lower bound on the average length of the shortest description of the random variable. For each event x, we can encode it by l_x bits. Then, the expected length of the description of X is $\sum_x p(x) l_x$. Gibbs' inequality shows that when $l_x = -\log_2 p(x)$, we have the shortest expected length. Specifically, the inequality states $-\sum_x p(x) \log p(x) \leq -\sum_x p(x) \log q(x)$ for any description with length $l_x = -\log q(x)$. It's reasonable to use the shortest expected length of description as a measure of information, since a process with more information inevitably requires more bits to describe.

Definition 1.2 (Joint Entropy and Conditional Entropy)**.** The joint entropy of a pair of discrete variables (X, Y) with joint distribution $p(x, y)$ is defined as

$$H(X, Y) = -\sum_{x} \sum_{y} p(x, y) \log p(x, y)$$

The conditional entropy $H(y|X)$ is then defined as

$$\begin{aligned} H(Y|X) &= -\sum_{x} p(x) \sum_{y} p(y|x) \log p(y|x) \\ &= \sum_{x} p(x) H(Y|X = x) \end{aligned}$$

Following the chain rule of probability, we also have the chain rule for entropy:

Theorem 1 (Chain Rule)**.**

$$H(X,Y) = H(X) + H(Y|X)$$

Definition 1.3 (Relative Entropy / Kullback–Leibler(KL) Divergence)**.** The relative entropy between two probability mass functions $p(x)$ and $q(x)$ is

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

KL Divergence measures the dissimilarity between two probability mass functions. However, KL Divergence is not a distance measure since it's not symmetric.

Theorem 2.

$$D(p||q) \geq 0$$

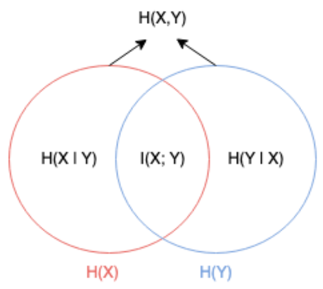
and

$$D(p||q) = 0 \textit{ if and only if } p = q$$

Definition 1.4 (Mutual Information)**.** The mutual information between two random variables X, Y is

$$\begin{aligned} I(X;Y) &= \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= D(p(x,y)||p(x)p(y)) \\ &= H(X) + H(Y) - H(X,Y) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

Mutual information measures the amount of information that one variable contains about another. This measure is symmetric.



This image demonstrates the relationship between mutual information and entropy. $H(X,Y)$ can be viewed as the information contained in joint (X,Y). $H(X,Y) \leq H(X) + H(Y)$ since X and Y may have some shared information $I(X,Y)$. $H(X|Y)$ measures the amount of additional information we get from X if we already know Y, so we need to remove the amount of shared information: $H(X|Y) = H(X) - I(X;Y)$.

Theorem 3.

$$I(X;Y) \geq 0$$

and

$$I(X;Y) = 0 \text{ if and only if } X \text{ and } Y \text{ are independent}$$

Theorem 4 (Data-processing Inequality)**.** *If we have of Markov chain of three random variables $X \to Y \to Z$, i.e., Z is conditionally independent from X given Y, then*

$$I(X;Y) \geq I(X;Z)$$

Remark. This is a really cool theorem! There is no data processing $Y \to Z$ such that we can get more information about X. Consider image classification as an example. Let X be the label of images. Let Y be images generated from the labels. And let f be a neural network converting Y into some embedding $Z = f(Y)$. We have a Markov chain $X \to Y \to f(Y)$. And based on this theorem, we know that $I(X;Y) \geq I(X;f(Y))$. It means that neural networks are not processing images to gain more information about Y! So what are neural network actually doing from an information-theoretic view? We will soon talk about information bottleneck which answers the question to some extent.

Definition 1.5 (Sufficient Statistics)**.** Given a family of probability mass functions $\{f_\theta(x)\}$. A statistic $T(X)$ is a function of the sample X from $\{f_\theta(x)\}$, e.g., sample mean or sample variance. We have a Markov chain $\theta \to X \to T(X)$. By Data-processing Inequality, we have

$$I(\theta, T(X)) \leq I(\theta, X)$$

$T(X)$ is called sufficient statistic if it contains all the information that X has about θ, i.e.,

$$I(\theta, T(X)) = I(\theta, X)$$

Remark. Taking the same example of classification. If the neural network f is a sufficient statistic, then we would have $I(X;Y) = I(X;f(Y))$. It means the embedding $f(Y)$ and the original image Y contain the same amount of information about label X.

2 Information Bottleneck in Deep Learning

In a typical supervised learning problem, we use input data from X to predict output label from Y. Ideally, we want the neural network to learn some embedding $\hat{X}$ such that all the information relevant to Y is kept and other information is discarded. We consider the Markov chain $Y \to X \to \hat{X}$. To discard

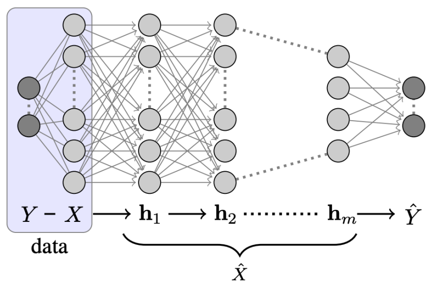


Figure 1: Source: Deep Learning and the Information Bottleneck Principle

irrelevant information, we want to maximally compress X so that only information about Y is kept. However, for a lossy compression, by Data-processing Inequality, $I(Y, \hat{X}) \leq I(Y, X)$. It means that $\hat{X}$ will loss information about Y. Obviously, there is a trade-off between compressing the representation and preserving meaningful information about Y. Information bottleneck method is a principle designed to extract relevant information in X about Y. It's called "information bottleneck" because we are passing the information in X about Y through a "bottleneck" (e.g., a neural network) to form a more compact representation $\hat{X}$.

Definition 2.1 (Information Bottleneck Principle)**.** To find a optimal representation $\hat{X}$, we want to minimize the functional

$$I(\hat{X}, X) - \beta I(\hat{X}, Y)$$

Remark.

1. Minimizing $I(\hat{X}, X)$ is maximizing compression. Without compression ($\hat{X}$ has the same information as X), we have $I(\hat{X}, X) = I(X, X) = H(X)$. With maximal compression ($\hat{X}$ only contains information of Y), we have $I(\hat{X}, X) = I(Y, X)$. And $I(Y, X) = H(X) - H(X|Y) \leq H(X)$.

2. Maximizing $I(\hat{X}, Y)$ is maximizing the relevant information in $\hat{X}$ about Y. This is obvious from the definition of mutual information.

3. β is the Lagrange multiplier. If $\beta = 0$, X will be maximally compressed at the risk of losing a lot of relevant information about Y. If $\beta \to \infty$, $\hat{X}$ would be a super detailed representation of X without any compression.

Now, if we consider an actual neural network as follow: This network forms the Markov chain $Y \to X \to h_1 \to \cdots \to h_m \to \hat{Y}$. By Data-processing inequality, we have

$$H(X) = I(X, X) \geq I(X, h_1) \geq \cdots \geq I(X, h_m) \geq I(X, \hat{Y})$$

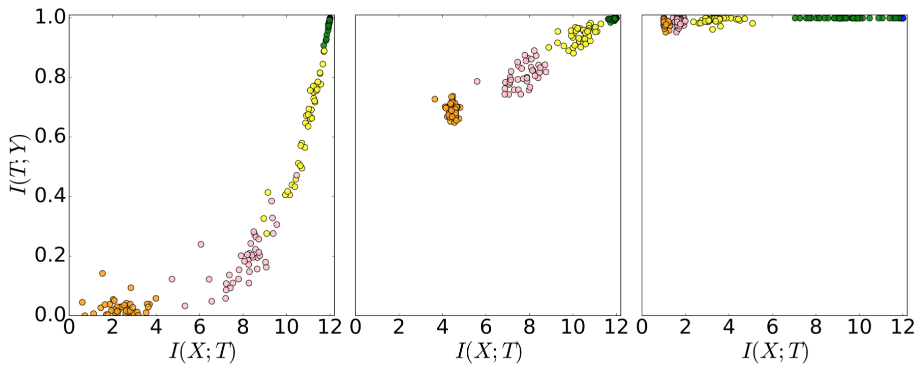
and

$$I(Y, X) \geq I(Y, h_1) \geq \cdots \geq I(Y, h_m) \geq I(Y, \hat{Y})$$

By Information Bottleneck Principle, we want to maximize $I(Y, \hat{Y})$ while minimizing $I(X, \hat{Y})$. For each layer, we want to maximize $I(Y, h_i)$ while minimizing $I(h_{i-1}, h_i)$.

Theorem 5 (Information Plane)**.** *Most supervise learning algorithms try to balance the trade-off between the performance on sampled data and the generalization to unseen data. Naftali Tishby argues that it's reasonable to consider $I(Y,T)$ as a measure of performance and consider $I(X,T)$ as a control of generalization.*

2.1 Optimization Phases of Neural Network



Source: OPENING THE BLACK BOX OF DEEP NEURAL NETWORKS VIA INFORMATION.

2.2 Input Compression Bound

Definition 2.2 (Vapnik-Chernovenkis (VC) bound)**.** As learned in lecture, the bound is

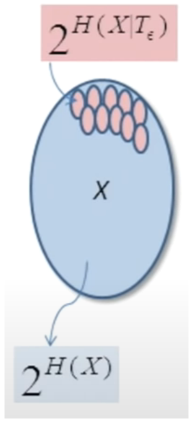
$$R(f_{ERM}) \leq R(f^*) + 2\sqrt{\frac{1}{2n} \log \frac{4|\mathcal{F}|}{\delta}}$$

where n is the number of samples, δ is the confidence, and $|\mathcal{F}|$ is the size of the hypothesis class. This bound becomes vacuous for neural networks which have super large $|\mathcal{F}|$.

Definition 2.3 (Input Compression Bound)**.** We want to get a bound that depends on the information of input instead of the large hypothesis class. The bound we will get is

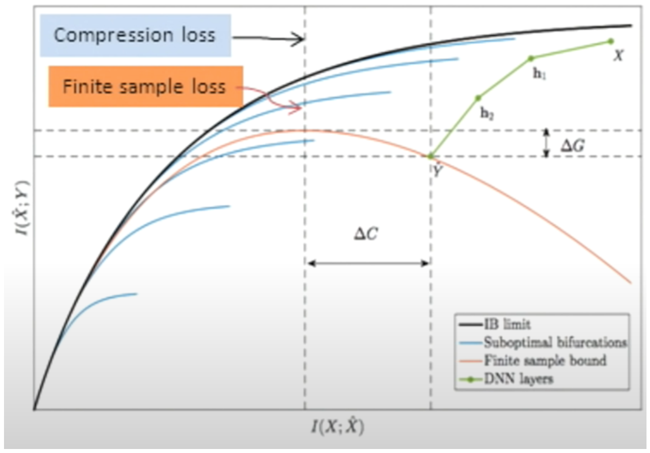
$$R(f_{ERM}) \leq R(f^*) + 2\sqrt{\frac{1}{2n}\left(2^{I(T_\epsilon, X)} + \log \frac{1}{\delta}\right)}$$

where T_ϵ is a ϵ partition of X. T_ϵ can be considered as embedding space where each X is compressed into some code. Note that if we can reduce $I(T_\epsilon, X)$ by k (k more bits of compression), we can reduce the training samples by a factor of 2^k!



This figure above shows the partition of the input space. We partition the input space X into small red balls $b_1, b_2, \cdots$. Each red ball can either be 0 or 1. Under this partition, each $x \in X$ has a more compact binary representation, e.g., $b_1 = 0, b_2 = 1, b_3 = 1, \cdots$. We can think of supervise learning as creating a good partition of X that aligns with the labels Y. The typical set of X has size $2^{H(X)}$ (you can read more in Elements of Information Theory 3.29). Similarly, each red ball has size $2^{H(X|T_\epsilon)}$. Then the size of T_ϵ (the number of red balls) is $|T_\epsilon| = \frac{2^{H(X)}}{2^{H(X|T_\epsilon)}} = 2^{I(X,T_\epsilon)}$. If we consider the hypothesis class of Boolean function that maps x to the binary representation $b_1 = 0, b_2 = 1, \cdots, b_{|T_\epsilon|}$, this class has size $2^{2^{I(X,T_\epsilon)}}$. Then, we can get the input compression bound from $\log 2^{2^{I(X,T_\epsilon)}}$.

2.3 $I(T,Y)$ and Performance



Source
If we know X, Y, we will have optimal lossy compression which is the black line. Then, the model's performance is subject to the compression loss. Since we only have finite data, the compression is worse and has the red line as the lower bound. The model's performance is subject to an additional finite sample loss.

3 References

1. A Mathematical Theory of Communication
2. Elements of Information Theory
3. Deep Learning and the Information Bottleneck Principle
4. Opening the black box of Deep Neural Networks via Information
5. Learning and generalization with the information bottleneck
6. A great unpublished note from Santosh Venkatesh on An Axiomatic Approach to Information

AWS Tutorial

ESE 5460

October 2023

Contents

1 Overview

The EC2 (Elastic Cloud Compute) service on AWS (https://aws.amazon.com) offers cloud computing infrastructure and provides access to GPU resources. Each student will be given compute credits to use on AWS. AWS could be used for the programming assignments and is a useful tool for the projects. EC2 gives you access to GPU instances which are charged by the hour. In this short tutorial, we will go over the basics, best practices and some useful tools

2 Why use the Cloud?

- Use latest GPU architectures
- Easier to scale
- No maintenance costs

3 Creating an Account

You will get an email titled "AWS @ Penn access" from isc-cloud-solutions@isc.upenn.edu with instructions on how to create your account and use the AWS credits allocated to ESE 546.

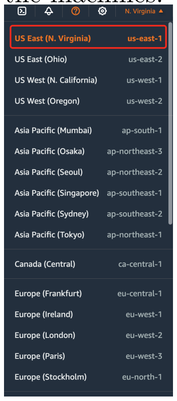
4 Launching a GPU instance

4.1 Use Penn Campus Network or University VPN

VPN Link

4.2 Switch your zone to Virginia (U.S East) on the top right of the screen.

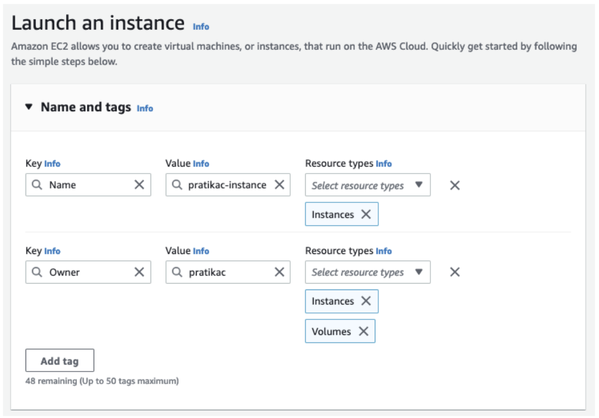
We are going to use this zone and it generally gives you better availability for the machines.



US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-1

4.3 Name and Tags

Search and go to the EC2 service in AWS and click on **Launch instance** to begin.


Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

▼ Name and tags Info

Key	Value	Resource types
Name	pratikac-instance	Instances
Owner	pratikac	Instances, Volumes

Add tag

48 remaining (Up to 50 tags maximum)

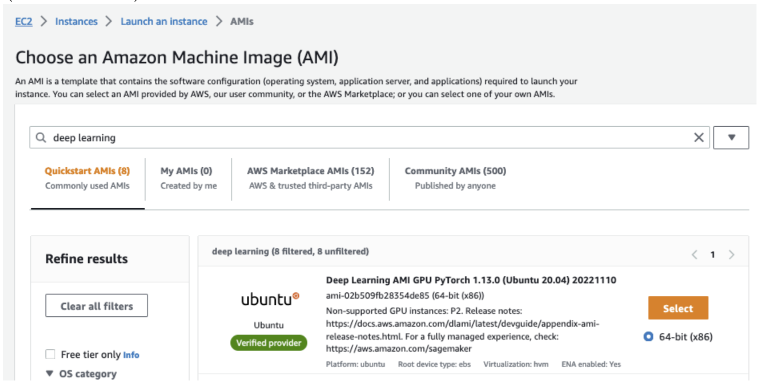
In Name and tags,

1. Give a name to your instance (it can be anything but it would be good to include your name e.g. pratikac-instance)

2. Click on **Additional Tags**, create a key called ”Owner” and value as ”your Penn email including everything after @” (i.e., pratikac@seas.upenn.edu for Pratik). Then add Intsances and Volumes to the Resource types. *We have scripts to prevent people from deleting each other’s instances accidentally. So you will not be able to interact with any instance or volume that was not launched with ”Owner:your-key”.*

4.4 Select Amazon Machine Image (AMI)

Under **Application and OS Images**, click **Browse more AMIs**, search **Deep Learning AMI**, and select “Deep Learning AMI GPU PyTorch ... (Ubuntu ...)”.


EC2 > Instances > Launch an instance > AMIs

Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

deep learning

Quickstart AMIs (8) Commonly used AMIs | My AMIs (0) Created by me | AWS Marketplace AMIs (152) AWS & trusted third-party AMIs | Community AMIs (500) Published by anyone

deep learning (8 filtered, 8 unfiltered)

Deep Learning AMI GPU PyTorch 1.13.0 (Ubuntu 20.04) 20221110
ami-02b509fb28354de85 (64-bit (x86))
Non-supported GPU instances: P2. Release notes: https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html. For a fully managed experience, check: https://aws.amazon.com/sagemaker
Platform: ubuntu Root device type: ebs Virtualization: hvm ENA enabled: Yes

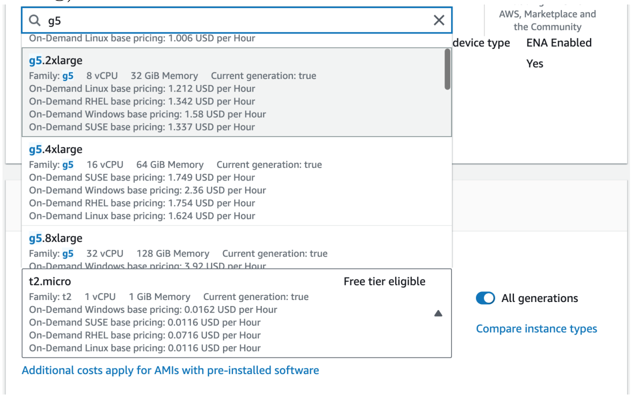
You can also use a standard Ubuntu but you will have to install everything (deep learning environment and libraries) yourself. so unless you know what you are doing, I would advise using the AMI.

4.5 Choose Instance Type

Under **Instance type**, change the instance type to one of the GPU instance. **You should use the following instances**:

1. g5.xlarge: 1 Nvidia A30 GPU (4 CPU cores, 16 GB memory). This should be sufficient for almost everyone and it is also the cheapest.
2. g5.2xlarge: 1 Nvidia A30 GPU (8 CPU cores, 32 GB memory). If you want run a dataloader with many threads, e.g., for lots of augmentation.
3. g5.12xlarge: 4 Nvidia A30 GPUs (48 CPU cores, 192 GB memory). Do not use this in general. Use it only if you need all 4 GPUs. This instance costs 12 dollars/hour so you will use your 100-150 dollars or so credits very quickly.
4. You can also use g4 instances and they are only a tiny bit slower (these have Nvidia T4 GPU which is the same one that Google Colab, so they are equally fast).

You can run multiple training runs simultaneously on the same GPU without changing any of your code. This is not different from running multiple processes on the same CPU. You do not need multiple GPUs just because you are training multiple neural networks. **Do not use the other g5 instances** unless you know for sure that you will benefit from them (and you know what you are doing).



g5

On-Demand Linux base pricing: 1.006 USD per Hour

g5.2xlarge
Family: g5   8 vCPU   32 GiB Memory   Current generation: true
On-Demand Linux base pricing: 1.212 USD per Hour
On-Demand RHEL base pricing: 1.342 USD per Hour
On-Demand Windows base pricing: 1.58 USD per Hour
On-Demand SUSE base pricing: 1.337 USD per Hour

g5.4xlarge
Family: g5   16 vCPU   64 GiB Memory   Current generation: true
On-Demand SUSE base pricing: 1.749 USD per Hour
On-Demand Windows base pricing: 2.36 USD per Hour
On-Demand RHEL base pricing: 1.754 USD per Hour
On-Demand Linux base pricing: 1.624 USD per Hour

g5.8xlarge
Family: g5   32 vCPU   128 GiB Memory   Current generation: true
On-Demand Windows base pricing: 3.92 USD per Hour

t2.micro   Free tier eligible
Family: t2   1 vCPU   1 GiB Memory   Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.0716 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

Additional costs apply for AMIs with pre-installed software

AWS, Marketplace and the Community

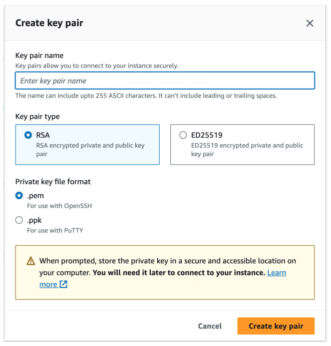
device type   ENA Enabled
Yes

All generations

Compare instance types

4.6 Create a keypair from the EC2 console

Under the Key pair (login) section, **Create new key pair**. Without this keypair you will not be able to SSH into the instance and will not be able to use it. It is a good idea to call the keypair by your Penn email, e.g., pratikac for Pratik.



Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

Enter key pair name

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair

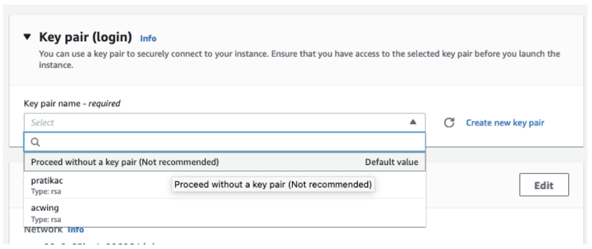
ED25519 — ED25519 encrypted private and public key pair

Private key file format

[✓ .pem] For use with OpenSSH

.ppk — For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more



Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

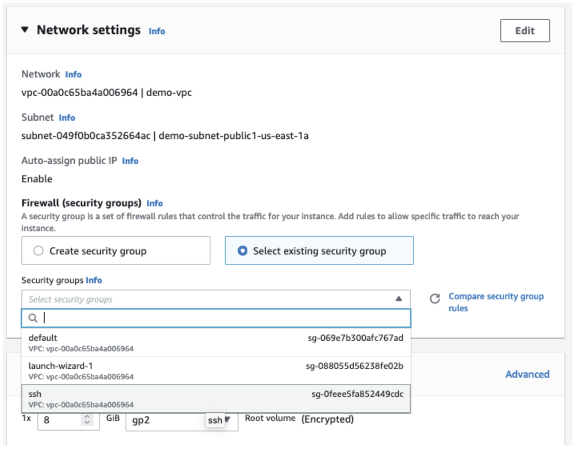
Select

Proceed without a key pair (Not recommended)	Default value
pratikac Type: rsa	
acwing Type: rsa	

Proceed without a key pair (Not recommended)

4.7 Security Groups

Under **Network Settings**, choose the option to **Select existing security group** and select the security group titled "ssh". Do not try to create a new one (you will not be able to). We have forwarded a few ports (they are: 60000-60100, 1024-1048, and 22) that you can use to connect to Jupyter, VScode, Tensorboard, etc. Make sure that the option "**Auto-assign public IP**" to set to enable as in some cases it might be set to disable by default (you might need to click the **edit** button on right top to see the option).



Network settings Info

Network: vpc-00a0c65ba4a006964 | demo-vpc

Subnet: subnet-049f0b0ca352664ac | demo-subnet-public1-us-east-1a

Auto-assign public IP: Enable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group [✓ Select existing security group]

Security groups Info

Select security groups

default VPC: vpc-00a0c65ba4a006964	sg-069e7b300afc767ad
launch-wizard-1 VPC: vpc-00a0c65ba4a006964	sg-088055d56238fe02b
ssh VPC: vpc-00a0c65ba4a006964	sg-0feee5fa852449cdc

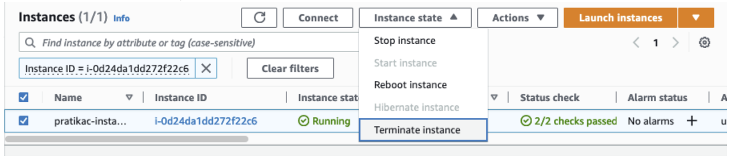
1x 8 GiB gp2 Root volume (Encrypted)

4.8 Launch the Instance

Click the **Launch Instance** button and you have a launched GPU instance to play with!

4.9 Stop and Start Instance

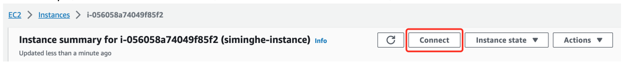
We have created a script that shuts down idle instances after 1 hour. If you see that your instance has "stopped" you can just start it using the button at the top. **Note: Always stop your instance after use.** Although the provided credits should suffice for this course, it is important to use resources carefully. An unstopped instance can quickly deplete the credits. **Stopping** the instance will not delete your data. **Terminating** the instance will delete all your data.



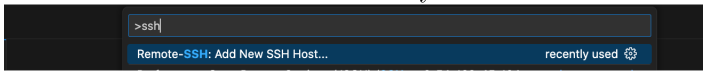
5 Using Your Instance

5.1 Use Your Instance in VS Code

Click **Connect** and you will be directed to a page called **Connect to instance**. Choose the **SSH Client** tab. You will see an example: "ssh -i 'your-pennkey.pem' address.com". You can use this to access your instance from your laptop. **Important**: you need to change 'your-pennkey.pem' to '/absolute-path/your-pennkey.pem' so that your computer can find the file.


EC2 > Instances > i-056058a74049f85f2
Instance summary for i-056058a74049f85f2 (siminghe-instance) Info
Updated less than a minute ago
Connect
Instance state
Actions

Go to vs code, press **command + shift + P**, type in ssh, and select **add new SSH Host**. Then type in "ssh -i '/absolute-path/your-pennkey.pem' address.com". You will be connect to your instance.



5.2 Check the deep learning environment

You can check your environment by opening an terminal in the vs code window and type

```
conda info --env
```

You can activate the environment you want, e.g., pytorch, by

```
conda activate pytorch
```

and run your code by

```
python file-name.py
```

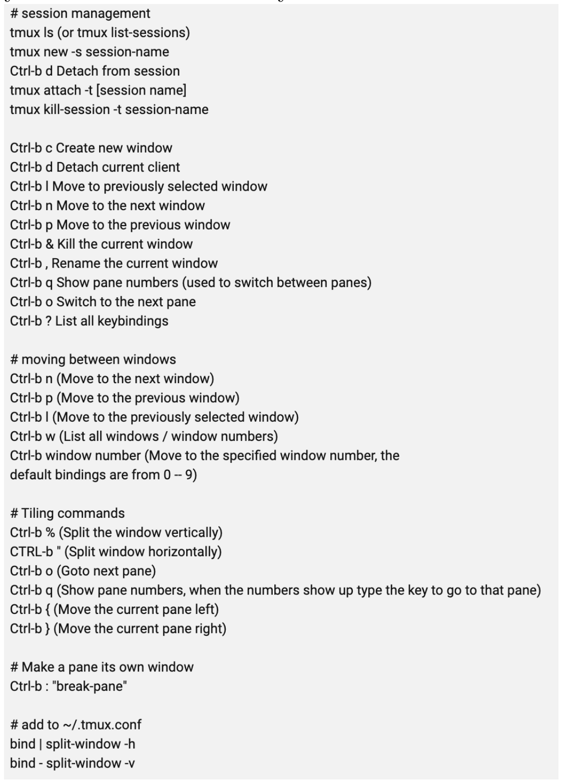
5.3 SCP: Secure Copy for File Transfering

You can use the following command to copy file from your computer to your AWS instance and vice versa. You can search for more details online.

```
scp -i '/absolute-path/penn-key.pem' source_path_of_file destination_path_of_file
```

5.4 tmux

Training may take a long time. If you accidentally closed the terminal or vs code, the terminal process will be closed which means your training will be killed and lost. You don't want this to happen! If you use tools like tmux, even if you close your terminal, the process will still be ran in the background and you can access it when you come back.Basic commands:



```
# session management
tmux ls (or tmux list-sessions)
tmux new -s session-name
Ctrl-b d Detach from session
tmux attach -t [session name]
tmux kill-session -t session-name

Ctrl-b c Create new window
Ctrl-b d Detach current client
Ctrl-b l Move to previously selected window
Ctrl-b n Move to the next window
Ctrl-b p Move to the previous window
Ctrl-b & Kill the current window
Ctrl-b , Rename the current window
Ctrl-b q Show pane numbers (used to switch between panes)
Ctrl-b o Switch to the next pane
Ctrl-b ? List all keybindings

# moving between windows
Ctrl-b n (Move to the next window)
Ctrl-b p (Move to the previous window)
Ctrl-b l (Move to the previously selected window)
Ctrl-b w (List all windows / window numbers)
Ctrl-b window number (Move to the specified window number, the
default bindings are from 0 -- 9)

# Tiling commands
Ctrl-b % (Split the window vertically)
CTRL-b " (Split window horizontally)
Ctrl-b o (Goto next pane)
Ctrl-b q (Show pane numbers, when the numbers show up type the key to go to that pane)
Ctrl-b { (Move the current pane left)
Ctrl-b } (Move the current pane right)

# Make a pane its own window
Ctrl-b : "break-pane"

# add to ~/.tmux.conf
bind | split-window -h
bind - split-window -v
```

5.5 Use Your Instance in Jupyter Notebook

Follow this tutorial! Link