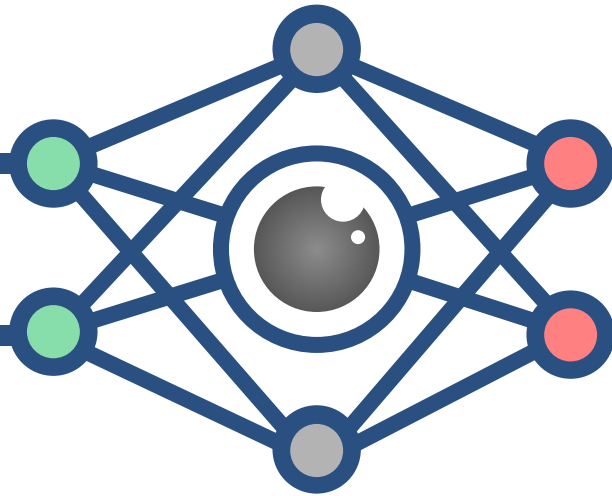


CS3485

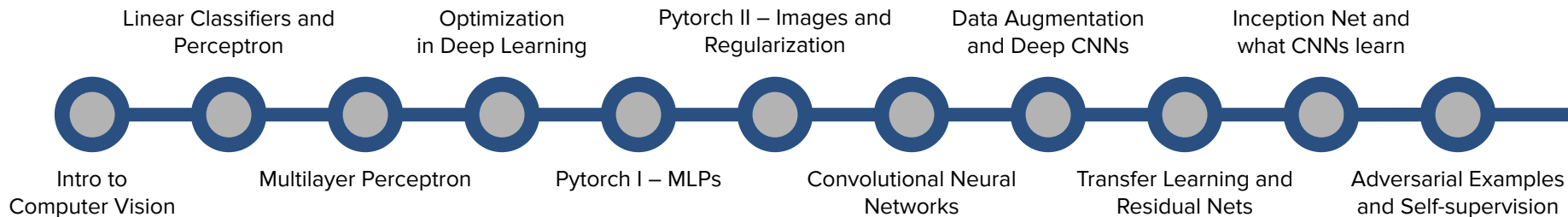
Deep Learning for Computer Vision



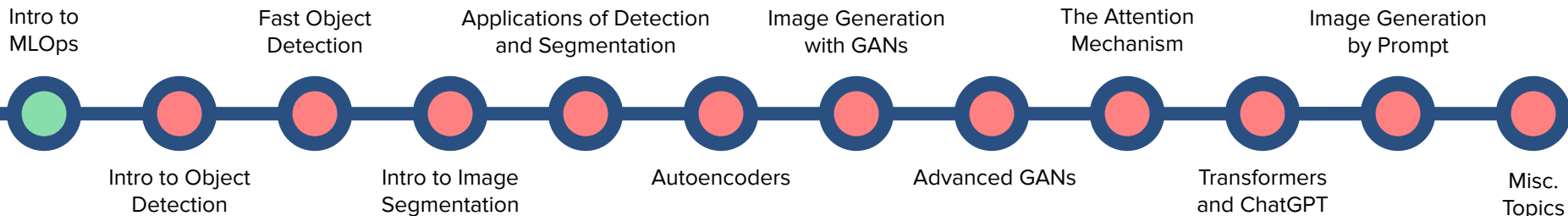
Lec 12: Intro to MLOps

(Tentative) Lecture Roadmap

Basics of Deep Learning



Deep Learning and Computer Vision in Practice



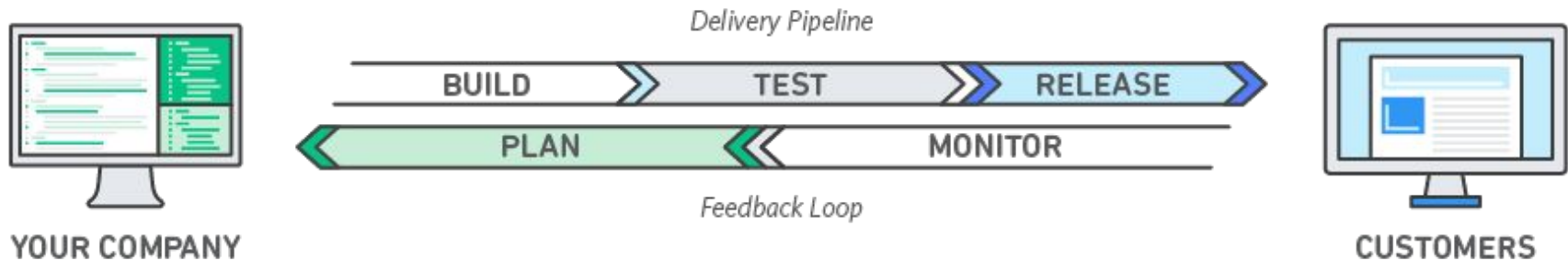
Deep Learning Theory vs Practice

- So far, we've seen some of the theory of Deep Learning and applied it to the problem of Image Classification.
- But obviously, most of Deep Learning's societal impact comes from its usage in the industry.
- Today, we'll dive a little bit on how these models are used in practice and how they are inserted in the software production line
- In order to know that, we'll dive into the world of **MLOps**.



A little digression: DevOps

- In any software company, there are two teams: **development** and **operations**:
 - The Development team plans, builds, tests new systems,
 - The Operations team implements and releases the products, and monitors user experience.
- These teams then create a **delivery pipeline** and a **feedback loop** specific to each team.
- However, often these teams may step on each other toes:
 - Sometimes, the operations team may provide feedback on bugs that need to be immediately fixed, causing delays in the development cycle.
 - Sometimes, the development “gets stuck” due to operational slowness.

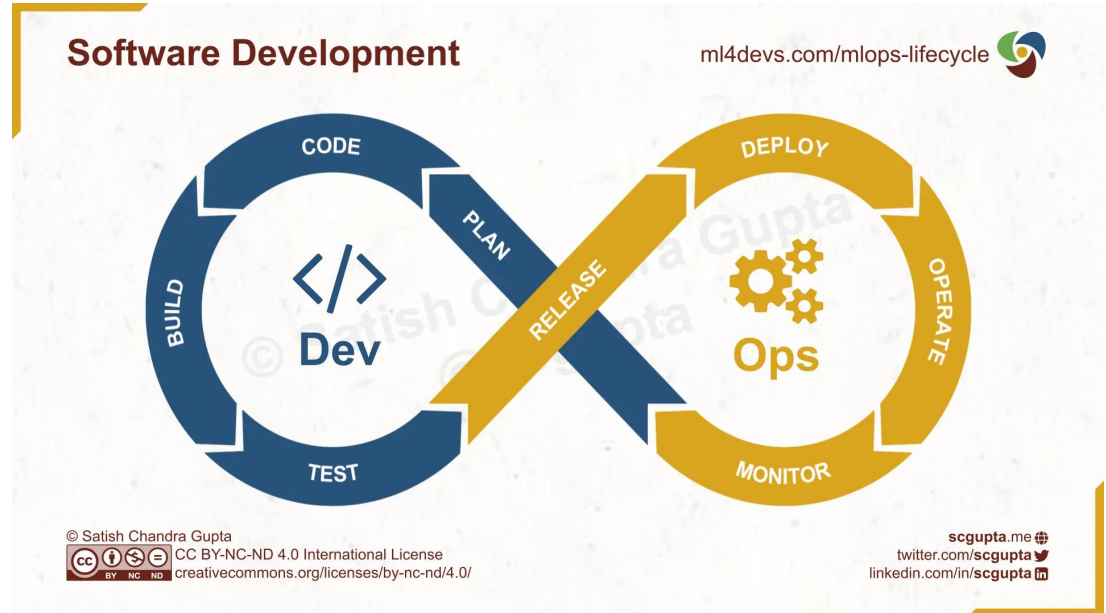


A little digression: DevOps

- **DevOps** unites Development and Operations teams, breaking down traditional silos between them and serves multiple purposes:
 - Faster launch of new features.
 - Increased customer and developer satisfaction through efficient processes,
 - Feedback loops for better communication.
- DevOps shifts the focus from team to **organizational goals**, from finger-pointing to **collective ownership**, enabling development and operations teams to work together seamlessly.
- It also enables the implementation of **CI/CD** processes:
 - **Continuous Integration (CI):** Developers frequently commit code changes that are automatically built, packaged, and tested through an automated pipeline.
 - **Continuous Delivery/Deployment (CD):** Code changes that have passed through the CI pipeline are automatically prepared for deployment and can be rapidly and safely released to production environments.

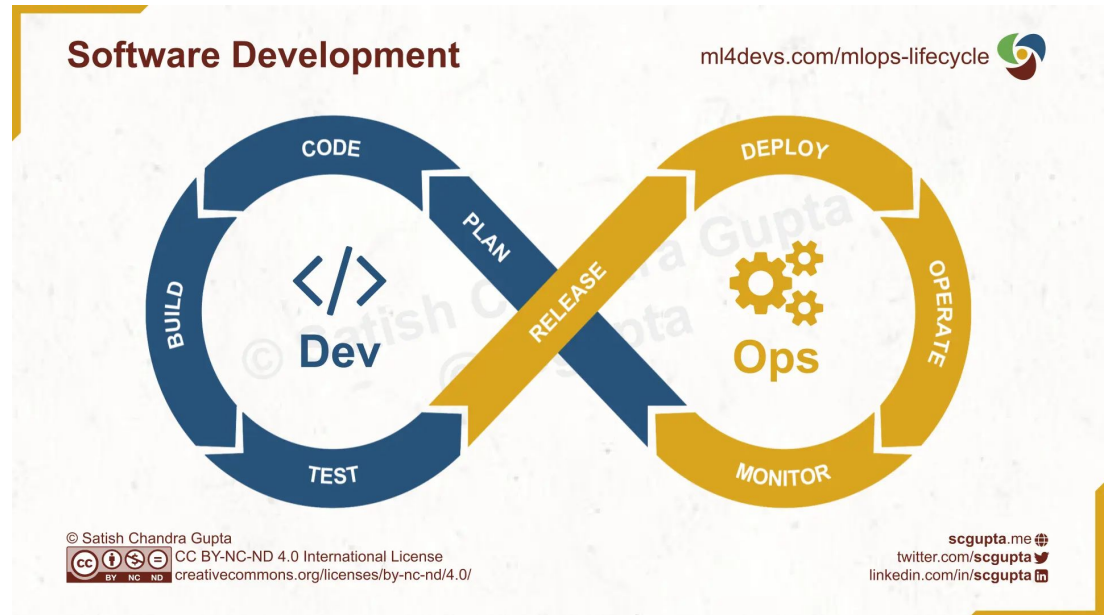
A little digression: DevOps LifeCycle

- In practice, companies implement the famous **DevOps lifecycle**, which consists of eight main stages represented as an infinity loop:
 - **Planning:** You collect the end-user data and create a roadmap of future processes at this stage.
 - **Coding:** At this stage, developers use tools or plugins to streamline the development process.
 - **Building:** Once developers finish coding, they commit the code to the shared repository.



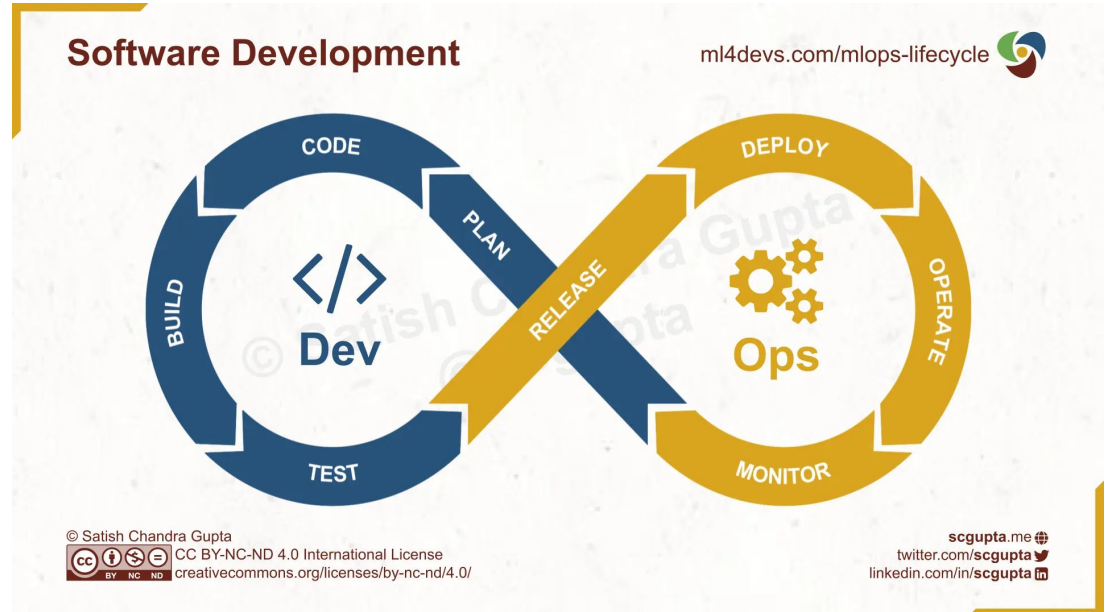
A little digression: DevOps LifeCycle

- In practice, companies implement the famous **DevOps lifecycle**, which consists of eight main stages represented as an infinity loop:
 - **Testing:** Look for bugs at all levels, i.e, unit tests, integration tests, coverage tests, performance tests, load tests, privacy tests, security tests, and bias tests.
 - **Releasing:** The DevOps team makes sure the build has passed all necessary tests in accordance with organisational needs.



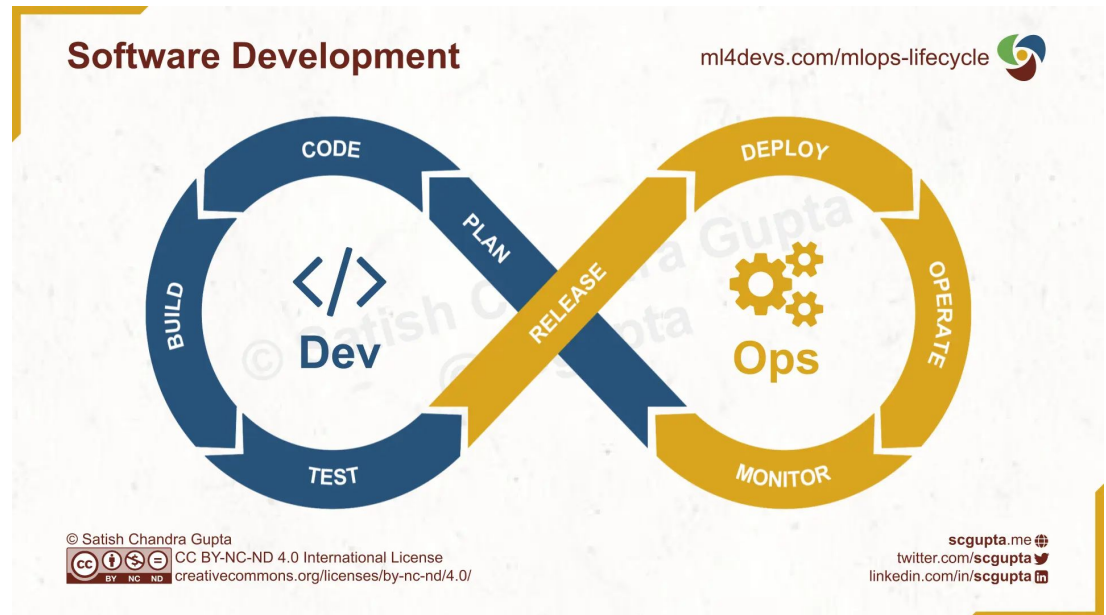
A little digression: DevOps LifeCycle

- In practice, companies implement the famous **DevOps lifecycle**, which consists of eight main stages represented as an infinity loop:
 - **Deployment.** You need to create the production environment and release the build at this stage. When deploying, we usually *create an endpoint* to serve our product.
 - **Operating.** Here, your product is ready for use by clients, and the operation team is still managing server configurations.



A little digression: DevOps LifeCycle

- In practice, companies implement the famous **DevOps lifecycle**, which consists of eight main stages represented as an infinity loop:
 - **Monitoring.** You can collect the data from customer experience and the first use of the product and use it for improving productivity and fixing possible bugs.
- This process then repeats and cycles continuously, as bugs are fixed, and company priorities change, without impacting product quality and processes.



MLOps

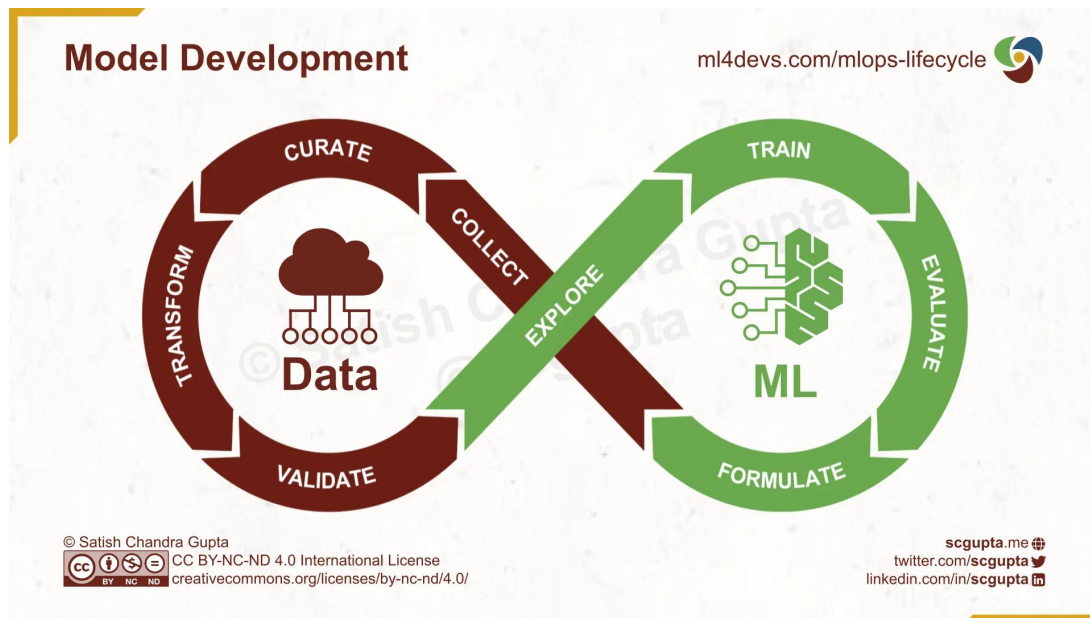
- With the rise in importance of Machine Learning (and Deep Learning in particular) in the software industry, typical ML pipelines started being incorporated in DevOps.
- Indeed, ML problems are found in many industrial sectors. For example:
 - In **finance**, software may to handle fraud detection, risk management and algorithmic trading.
 - In **manufacturing**, there is supply chain optimization, demand forecasting and quality control.
 - In **media and entertainment**, we find content recommendation and sentiment analysis.
- That gave rise to what is called **MLOps** to streamline and enhance the deployment, monitoring, and management of machine learning models
- The field of MLOps is not yet as established as DevOps, but many companies are already **eager to hire** professionals in that ML/Dev/Ops intersection.



A typical MLOps engineer salary according to [Glassdoor](#).

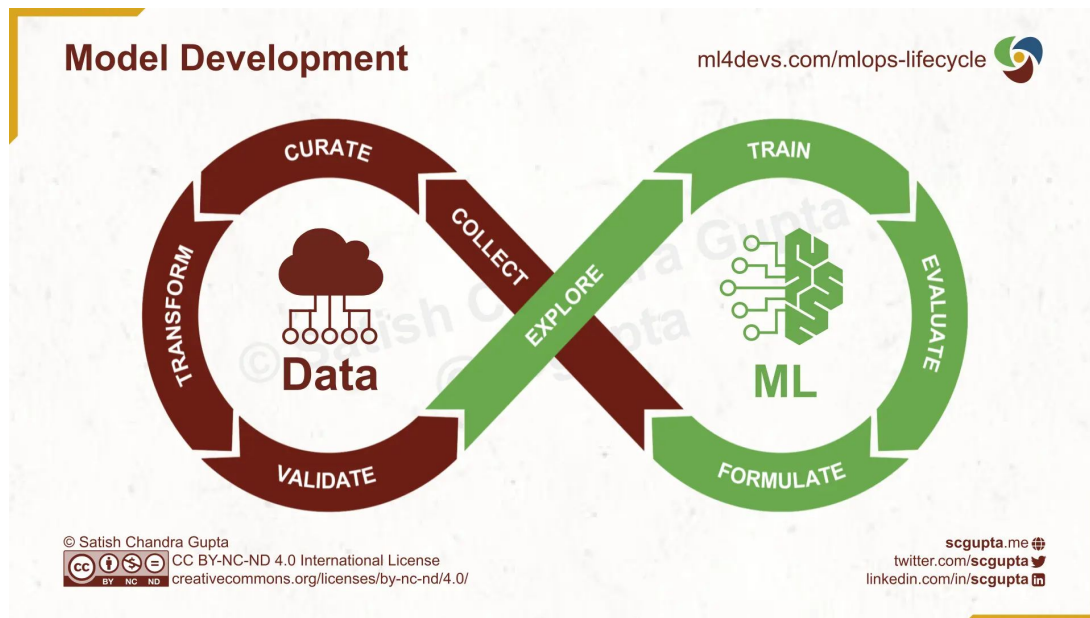
MLOps Lifecycle

- Just like with DevOps, MLOps also has a lifecycle that consists of many data driven stages. One example of such has the following stages:
 - **Formulate:** Translating a business objective into a machine learning problem. Some factors to consider:
 - a. *Cost of mistakes:* How costly are model mistakes?
 - b. *Data:* is the data available/cheap?
 - c. *Evaluation Metrics:* What metric do you want to optimize?



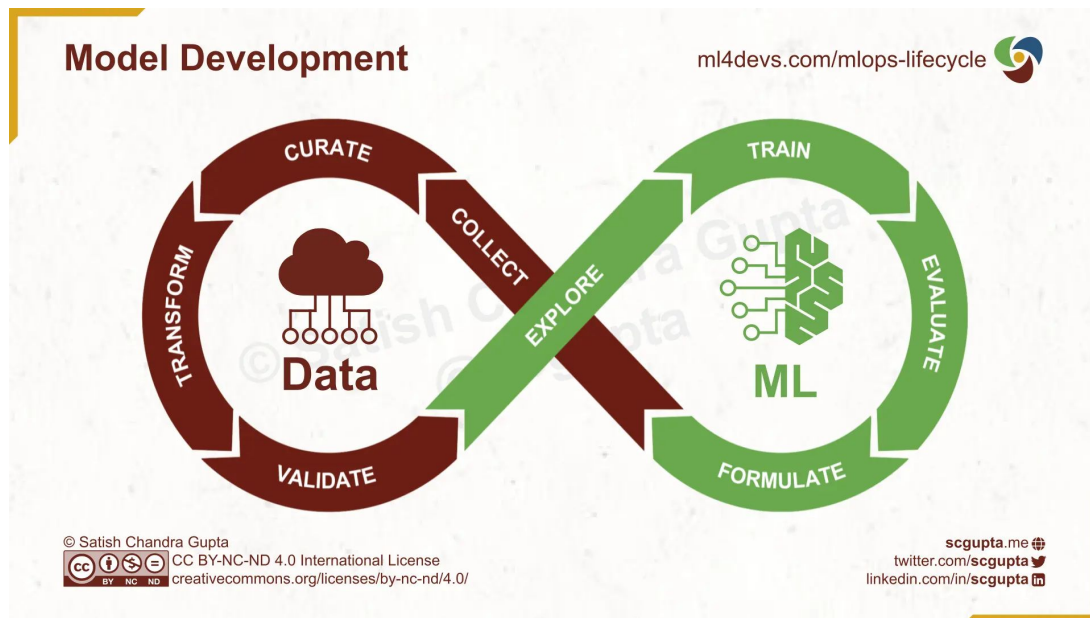
MLOps Lifecycle

- Just like with DevOps, MLOps also has a lifecycle that consists of many data driven stages. One example of such has the following stages:
 - **Collect:** collect the necessary data from internal applications as well as external sources.
 - **Curate:** Collected data is almost never pristine. You need clean it, remove duplicates, fill in missing values, and store it.
 - **Transform:** You transform the cleaned data to suit your ML modeling.



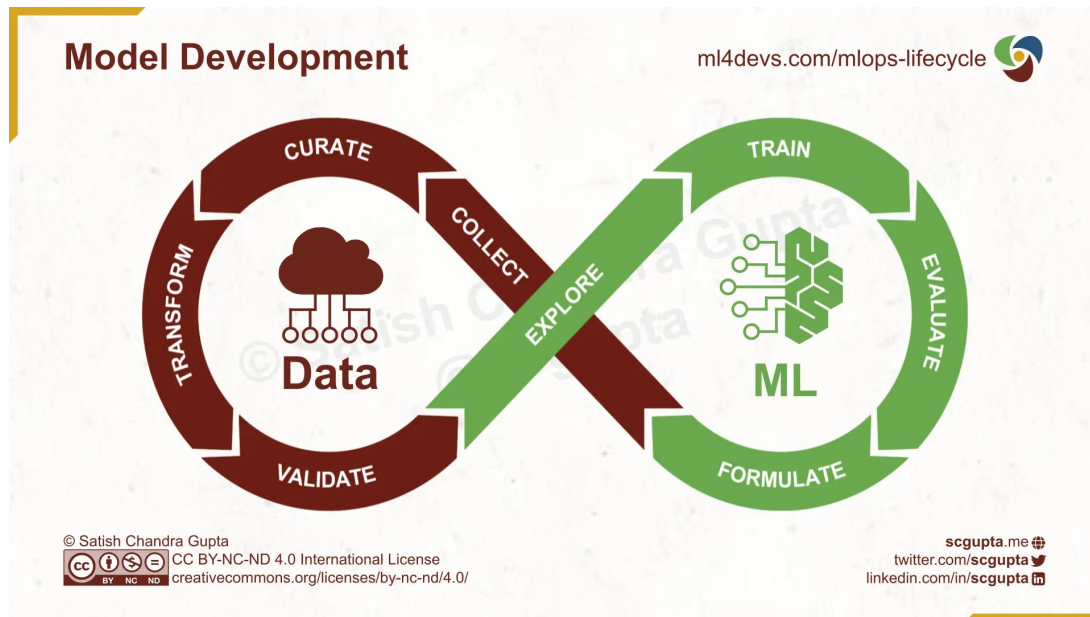
MLOps Lifecycle

- Just like with DevOps, MLOps also has a lifecycle that consists of many data driven stages. One example of such has the following stages:
 - **Validate:** Implement quality checks, maintain logs of data distributions over time, and create triggers to alert when any of the checks fail.
 - **Explore:** ML engineers perform data analysis to understand the relationships between various features and the target values.



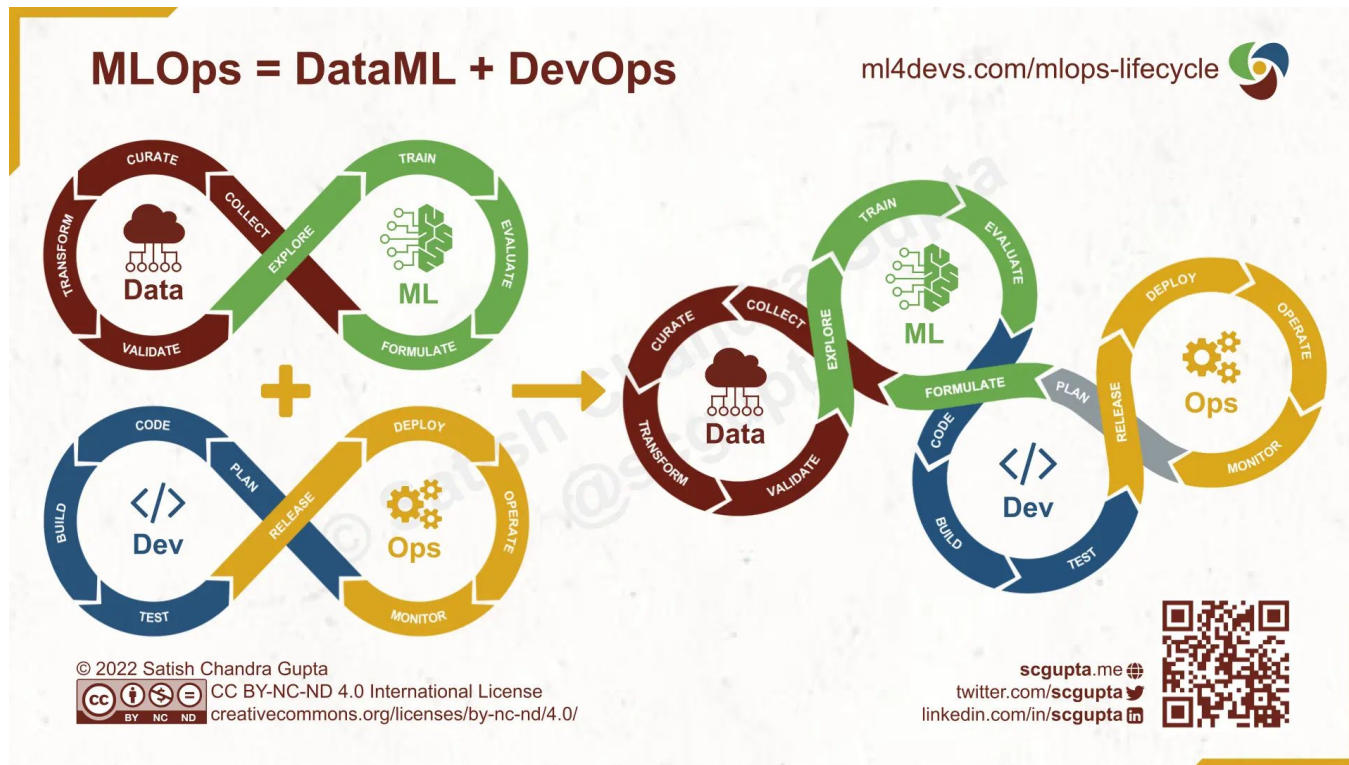
MLOps Lifecycle

- Just like with DevOps, MLOps also has a lifecycle that consists of many data driven stages. One example of such has the following stages:
 - **Train:** ML engineers train multiple models, run experiments, compare performances, tune hyper-parameters, and select a couple of best-performing models.
 - **Evaluate:** Evaluate the model characteristics against business objectives and pre-established metrics.



DevOps and MLOps

- In practice, ML model and software development are becoming more and more tied together into one larger eternal **MLOps knot** (which some define as the true MLOps lifecycle).



Example of deployment of an ML model

- The connection between model training and deployment is crucial, and we won't have a lot of time in this course to see how this happens in practice.
- However, I can at least show you a bit what a simple deployment of one of our previous codes looks like.
- Here, we'll use the **Cat vs. Dog classification problem**, solved using transfer learning on a small network architecture called [MobileNet](#) implemented in Pytorch.
- For deployment, we'll use a Python library called **Flask**, a web framework that allows developers to build simple (and also complex) web applications very easily.
- Let's see how it works in practice (and here's the code if you want to try it yourself)

Pytorch Code
(for training the model)

Flask Code
(for deploying the model)

[Click here to open code in Colab](#)



[Click here to open code in Github](#)



Example of deployment of an ML model

Cat vs. Dog Classifier

Upload Image

Predict the Image



Classification Results

This is a **cat** (99.58% confidence)!

Cat vs. Dog Classifier

Upload Image

Predict the Image



Classification Results

This is a **dog** (99.2% confidence)!

Cat vs. Dog Classifier

Upload Image

Predict the Image



Classification Results

This is a **other** (74.35% confidence)!

Exercise (in pairs)

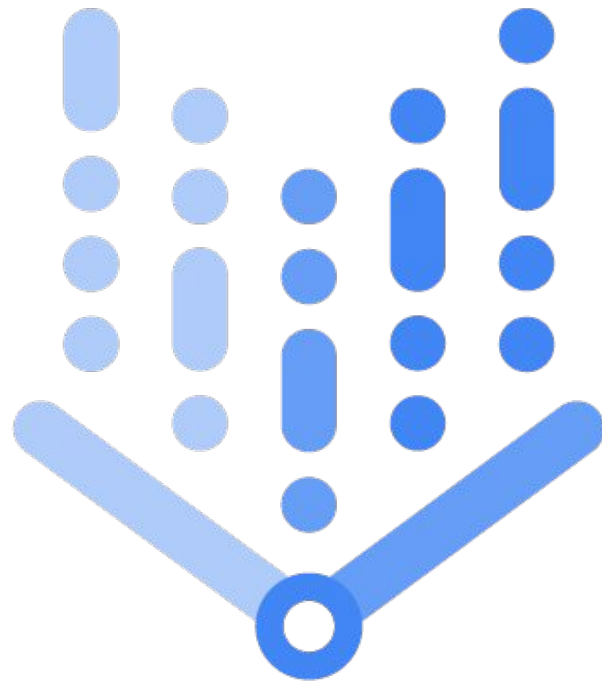
- Think of an industry and a particular task within that industry may take advantage of using ML in its processes. Then design an MLOps pipeline (the *formulate* step, in particular, but you can brush the others as well) that continuously improve and maintains that pipeline. *For example*: say I want to predict how good my lectures are using ML and use it to improve future classes. Then:
 - *Formulate*: ML mistakes are not so costly, I can use my slides or recordings as input data and students evaluations as output data, the metric could be whether I reached 70% of overall satisfaction.
 - *Train*: Each time, at the end of the semester, my software automatically loads my new slides and recordings and retrains the prediction network.
 - *Evaluate*: I can then test my newly trained model on data from years where I know the class was successful and those where it wasn't.

A case study: Vertex AI

- There are also specific tools that can be used solely when automatizing MLOps processes.
- Nowadays, the most useful ones are embedded in Machine Learning platforms with the biggest cloud computing providers*:
 - Microsoft's Azure provides **Azure ML**
 - Amazon's AWS (Amazon Web Services) provides **SageMaker**
 - Google's GCP (Google Cloud Platform) provides **Vertex AI**.
- In this course, we'll focus our attention to Vertex AI as its platform is in some ways similar to Colab's**.

* Although some [smaller companies](#) are also starting to take a larger share of that market.

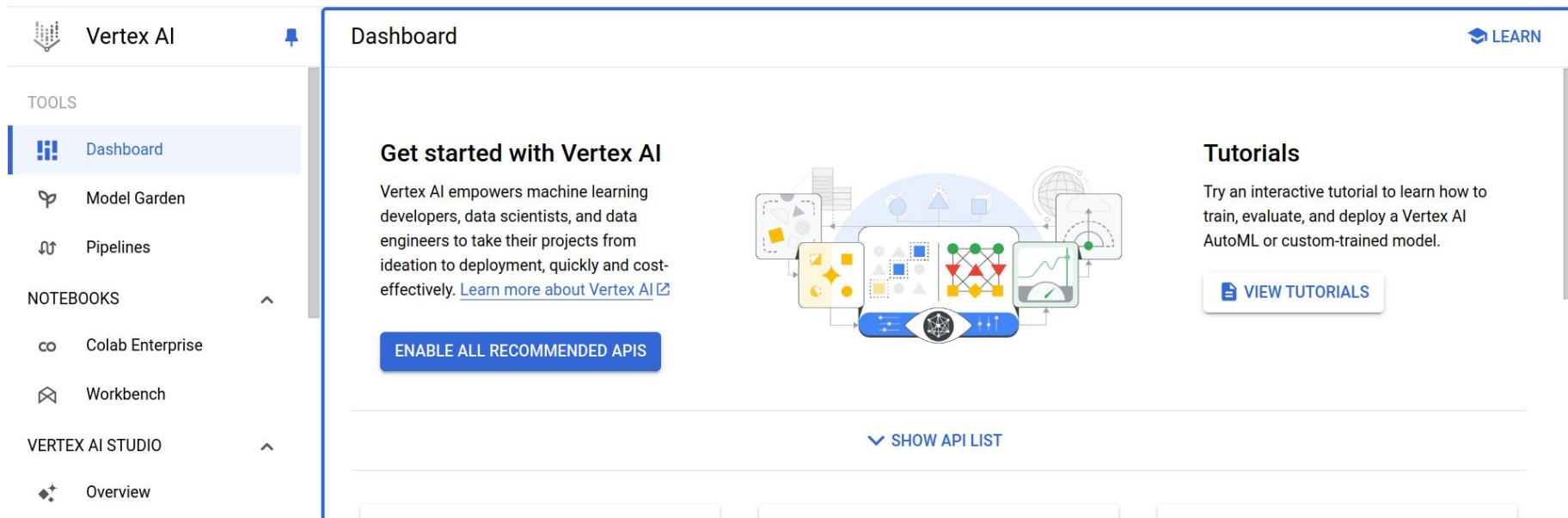
** [Here](#)'s a good summary of Vertex AI's tools for MLOps.



Google's GCP Vertex AI

A case study: Vertex AI

- After logging in GCP, we can hit “console” for some project and search for Vertex AI. We’ll then see the following dashboard:



The screenshot displays the Vertex AI dashboard interface. On the left is a navigation sidebar with sections for TOOLS (Dashboard, Model Garden, Pipelines), NOTEBOOKS (Colab Enterprise, Workbench), and VERTEX AI STUDIO (Overview). The main content area is titled 'Dashboard' and features a 'LEARN' button in the top right. The primary section is 'Get started with Vertex AI', which includes a descriptive paragraph and a blue button labeled 'ENABLE ALL RECOMMENDED APIS'. To the right of this text is a central graphic showing a data pipeline with various icons like a bar chart, neural network, and globe. Further right is a 'Tutorials' section with a paragraph and a 'VIEW TUTORIALS' button. At the bottom of the main content area, there is a 'SHOW API LIST' button.

Vertex AI

Dashboard

LEARN

TOOLS

- Dashboard
- Model Garden
- Pipelines

NOTEBOOKS

- Colab Enterprise
- Workbench

VERTEX AI STUDIO

- Overview

Get started with Vertex AI

Vertex AI empowers machine learning developers, data scientists, and data engineers to take their projects from ideation to deployment, quickly and cost-effectively. [Learn more about Vertex AI](#)

ENABLE ALL RECOMMENDED APIS

Tutorials

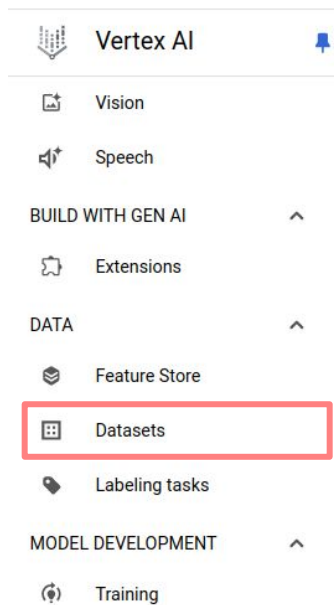
Try an interactive tutorial to learn how to train, evaluate, and deploy a Vertex AI AutoML or custom-trained model.

VIEW TUTORIALS

SHOW API LIST

A case study: Vertex AI

- At Vertex AI, you have tools to: **manage datasets**



A case study: Vertex AI

- At Vertex AI, you have tools to: manage datasets, **use pretrained vision (or language) models**

The image displays two screenshots of the Vertex AI interface. The left screenshot shows the main navigation menu with the following items: Vision, Speech, BUILD WITH GEN AI, Extensions, DATA, Feature Store, Datasets (highlighted with a grey box), Labeling tasks, and MODEL DEVELOPMENT, Training. The right screenshot shows the 'Vertex AI Studio' sub-menu with the following items: Overview, Multimodal, Language, Vision (highlighted with a red box), Speech, BUILD WITH GEN AI, Extensions, DATA, and Feature Store.

A case study: Vertex AI

- At Vertex AI, you have tools to: manage datasets, use pretrained vision (or language) models, **train your own models**

The image displays three screenshots of the Vertex AI interface, illustrating different ways to access 'Colab Enterprise'.

- Left Screenshot:** Shows the main navigation menu. The 'Datasets' option is highlighted with a grey box.
- Middle Screenshot:** Shows the 'Vertex AI Studio' navigation pane. The 'Vision' option is highlighted with a grey box.
- Right Screenshot:** Shows the 'Vertex AI Studio' interface. The 'Colab Enterprise' option is highlighted with a red box.

A case study: Vertex AI

- At Vertex AI, you have tools to: manage datasets, use pretrained vision (or language) models, train your own models, **design experiments**

The image displays four screenshots of the Vertex AI interface, each showing a different section of the navigation menu. The first screenshot shows the 'Vertex AI' menu with 'Datasets' highlighted. The second screenshot shows the 'Vertex AI STUDIO' menu with 'Vision' highlighted. The third screenshot shows the 'Vertex AI' menu with 'Colab Enterprise' highlighted. The fourth screenshot shows the 'Vertex AI' menu with 'Experiments' highlighted.

Vertex AI	Vertex AI	Vertex AI	Vertex AI
Vision	VERTEX AI STUDIO	Pipelines	Labeling tasks
Speech	Overview	NOTEBOOKS	MODEL DEVELOPMENT
BUILD WITH GEN AI	Multimodal	Colab Enterprise	Training
Extensions	Language	Workbench	Experiments
DATA	Vision	VERTEX AI STUDIO	Metadata
Feature Store	Speech	Overview	DEPLOY AND USE
Datasets	BUILD WITH GEN AI	Multimodal	Model Registry
Labeling tasks	Extensions	Language	Online prediction
MODEL DEVELOPMENT	DATA	Vision	Batch predictions
Training	Feature Store	Speech	Vector Search

A case study: Vertex AI

- At Vertex AI, you have tools to: manage datasets, use pretrained vision (or language) models, train your own models, design experiments, **deploy models to endpoints**, etc.

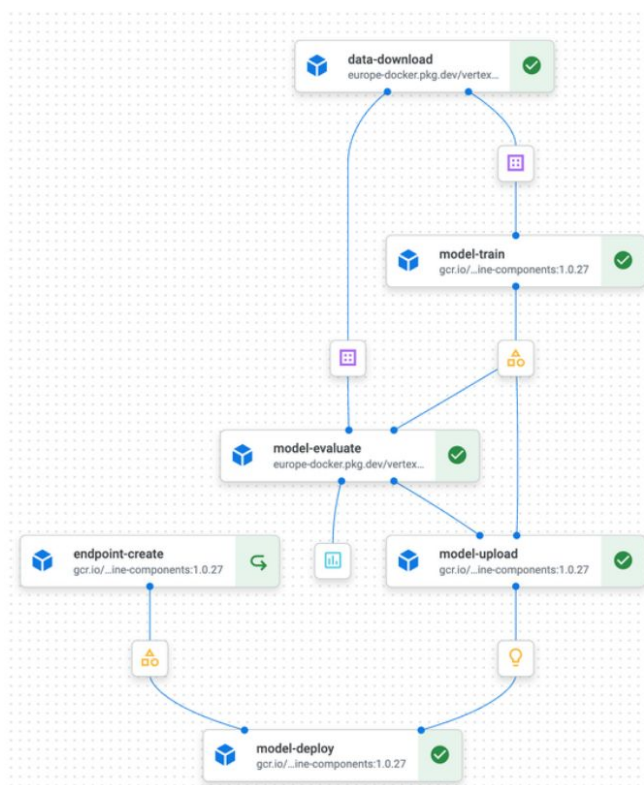
The image displays five sequential screenshots of the Vertex AI interface, illustrating various navigation paths to the 'Online prediction' feature. Each screenshot shows a different menu structure with specific items highlighted by a grey or red border.

- Screenshot 1:** Shows the main navigation menu with 'Datasets' highlighted by a grey border.
- Screenshot 2:** Shows the 'Vertex AI Studio' menu with 'Vision' highlighted by a grey border.
- Screenshot 3:** Shows the 'Vertex AI Studio' menu with 'Colab Enterprise' highlighted by a grey border.
- Screenshot 4:** Shows the 'Model Development' menu with 'Experiments' highlighted by a grey border.
- Screenshot 5:** Shows the 'Deploy and Use' menu with 'Online prediction' highlighted by a red border.

A case study: Vertex AI

- At the industrial level, developers **orchestrate** the whole MLOps life cycle in Vertex AI **programmatically** (i.e., via code, not the dashboard) using what are called Pipelines*.
- Here you have a pipeline created with Vertex AI's SDK (software development kit), which automatically:
 - Downloads data and trains the model with the data.
 - Evaluates and stores the model.
 - Deploys it to an API endpoint.
- Once you have the endpoint, you can connect it with any programmable services, from mobile and web apps to operating systems.

* [Here](#)'s a great video from this great channel called MikeStat that covers the whole Pipeline Orchestration in Vertex AI. The video also comes with a [great github repo](#).



Final Project Specifics

- One of your final project types this year will have an MLOps theme:
 - Pick a problem in an industry of your interest that can use Deep Learning in its processes.
 - Plan out how to implement each step of MLOps life cycle for that problem.
 - Implement some or all of those steps in practice.
 - Deploy the resulting model.
- A few notes here:
 - You'll can use any pretrained model you find online or in our slides (the focus won't be on the model *per se*).
 - You'll be mostly graded on the quality of your product.
 - Take this as something to put on your resume!



(Machine) Learning in an Ever Changing World

- If you choose that theme for your final project, you'll have to learn many new tools and concepts by yourself.
- This is somewhat intentional as well: **learning how to learn*** is essential to *any* intellectual endeavor and it is particularly crucial in modern software development.

It's Better To Know How
To Learn Than To Know.

-DR. SEUSS

* The following advices are from [Yonas Gebregziabher](#), a Bowdoin CS alumni who's currently working at Microsoft as a developer.

(Machine) Learning in an Ever Changing World

- There are a few reasons for why that is true in the industry:
 - *Technology evolves quickly:*
 - Frameworks get updates, new technologies are developed, and the industry often quickly responds to these changes.
 - You need to also **pick up on these changes** and be ready to implement them, should they significantly impact your product.
 - *Re-orgs are very common (especially in larger companies):*
 - Changes in management and shifts in business needs cause entire teams to abandon projects and reshuffle. This means you might find yourself on a new team with a tech stack you're not entirely familiar with.
 - You need to quickly **adapt to new teams** so that you can begin to contribute as well!
 - *The majority of a developer time is spent reading code not writing it:*
 - Oftentimes, you're maintaining an existing codebase, adding features, and fixing bugs.
 - You need to be able to develop skills to **comprehend code bases** so that you can quickly ramp up. [Here](#) is a good read on it.

How to learn how to learn?

- For these reasons, you need to be able to read code and read documentation often to learn new things. **Learning is at the core of a software developer's career.**
- But how do you learn? Here a few tips:
 - You need to figure out what works best for you:
 - Oftentimes, the best approach to learning something new is to simply just build something that incorporates whatever you want to learn.
 - You learn best by struggling to figure out a bug while working on a project that interests you.
 - It is completely okay to get started with YouTube videos that introduce you to these concepts,
 - However, it is counterproductive if you're copy-pasting everything you see in the videos all the time.
 - You'll get stuck in **tutorial hell**, where it's hard for you to reuse the technology you learned outside of the context of the video you followed. [Here](#) is a good read on tutorial hell and how to escape it.

How to learn how to learn?

- By building projects you find interesting, you do a couple of things:
 - a. You're more motivated to keep working on the project through the difficult stages. This keeps you wanting to learn more.
 - b. You learn truly what you're using to build your project because you've understood it enough to apply it to your use case.
 - c. You learn best by failing time and time again by working on stuff you find interesting.
- After years of this, picking up new things becomes easier because you've built a habit of learning that works best for you.



Video: AI Art Generation

