Deep Learning Foundations

Welcome to the *PracticumAI: Deep Learning Foundations* course! This course is intended to demystify the concepts of neural networks and deep learning. We will touch on how neural networks work, how to use them, and why neural networks have dominated AI research in the past decade. The first module of this course provides a high-level overview of deep learning and some hands-on experience with using these models. This course builds on the previous *Practicum AI* courses and will require some knowledge of Python, Jupyter Notebooks, and high-performance computing environments.

Module 1: Getting Started with Deep Learning

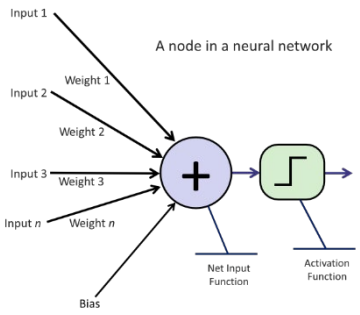
Module 1 Course Objectives:

By the end of this module, you will be able to:

1. Define a neural network.
2. Describe how a neural network works.
3. Discuss deep networks.
4. Discuss what can be done with neural networks.
5. Use a deep learning pre-trained model to classify an image.
6. Discuss Python AI Frameworks.

What Are Neural Networks?

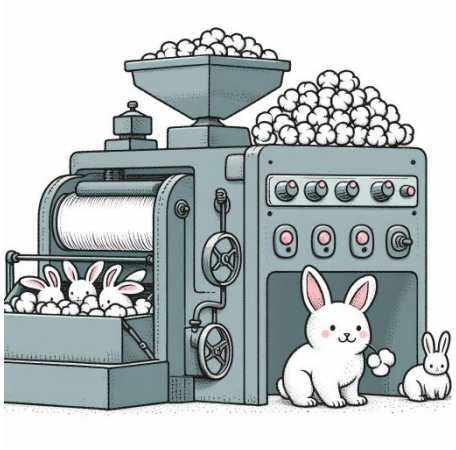
What exactly are neural networks and deep learning? The core of deep learning is the **neural network**. A neural network is a particular learning algorithm inspired by the billions of interconnected neurons in the human brain. It consists of interconnected **nodes** (analogous to neurons) that process information in layers. Each connection has a **weight**, which gets adjusted during training, allowing the network to "learn" from data and make predictions.



Module 2 will provide more details, but we will introduce some vocabulary here. Each input connection is assigned a weight for each node (or neuron). The input value is multiplied by the weight, and all these products are added together. Each node also has a **bias term** that is added to the sum. That sum is the raw output of the neuron. Neurons can have an **activation function**, which modifies the output before passing it on to the next node in the network. Activation functions enable non-linear functions to be learned, and which function to use is one of the **hyperparameters** or model settings you can set in training.

In a **deep learning** AI model, neurons are grouped in layers, as in a multi-layered cake. If you have many layers, the cake is "deep" when viewed from above. And that's why we say "deep learning."

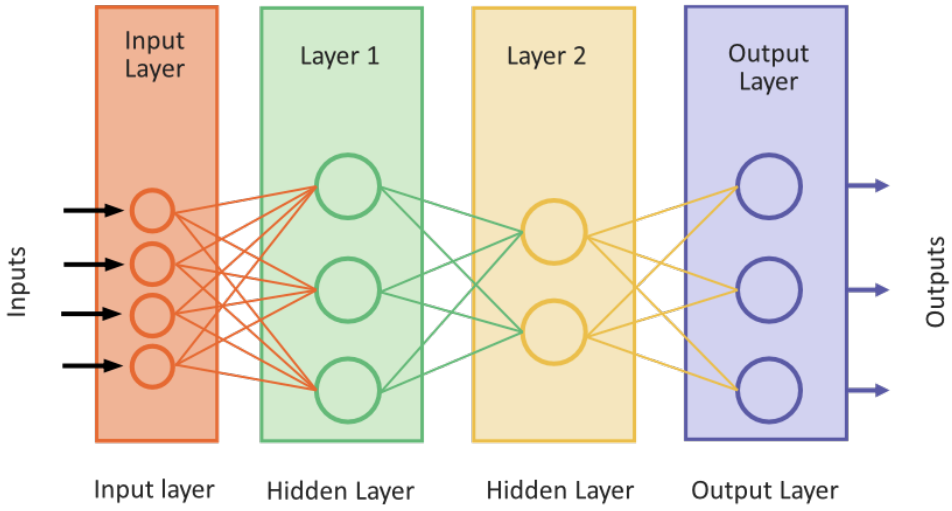
Deep Learning Foundations



As an analogy, imagine a factory assembly line that takes raw input materials and has several stations where workers perform some function to produce a product. There is an example product that the workers aim to replicate. Each station (a node or neuron) in our factory has a specific task. Raw materials (input data) are entered at the beginning, and each station makes slight modifications. By the time the materials reach the end of the assembly line, they've been transformed into a finished product (output or prediction). The product of the assembly line is then compared to the example product. The factory workers then adjust their tools or methods by comparing outputs to the ground truth to produce better products. Similarly, a neural network adjusts its weights based on the difference between its predictions and the actual outcomes, improving its accuracy over time.

A neural network is a series of interconnected nodes that aim to transform input data into meaningful output. In practice, there are typically many (thousands) of neurons in layers, and each layer takes input from the previous layer in the network and feeds its output to the next layer.

1. **Nodes or Neurons**: These are the fundamental units of a neural network. Each node receives information, processes it, and then sends it onward. We'll go deeper into how exactly nodes work in the next module.
2. **Layers**: A typical neural network is organized into three main types of layers:
    a. **Input Layer**: The first layer where data enters the system.
    b. **Hidden Layers**: These are layers that sit between the input and output layers. The data undergoes multiple transformations as it passes through these layers. There can be one or many hidden layers in a network. The term hidden layer refers to the fact that we don't see the details of how they operate—while we see the input going in and the output coming out, we don't directly see the operations in the middle.



    c. **Output Layer**: The final layer produces the result or prediction.
3. **Weights**: These are values associated with the connections between nodes. They determine the importance or influence of one node's output on another. During training, these weights are

adjusted to minimize the difference between the network's predictions and the actual outcomes.

4. **Activation Functions**: Each node in the network has an associated activation function. It decides how the information the node has processed should be transformed before being passed forward. Think of it as a filter that amplifies, diminishes, transforms, or maintains the signal it receives.

Training a neural network involves feeding it training data with known outcomes, having it make predictions, and then adjusting the weights based on how close or far its predictions are from the actual outcomes. This process is repeated many times, and the network becomes more adept with each iteration.

In essence, neural networks are like finely tuned machines that, over time, learn the best way to process information to achieve a desired outcome.

What is Deep Learning?

We've previously said that deep learning is a subfield of machine learning (both are subfields of artificial intelligence) and can be analogized to a layered cake. In keeping with our initial (and delicious) premise, here's a more thorough explanation of deep learning:

Imagine you're making a multi-layered cake, like a towering wedding cake. Each layer of the cake represents a layer in a neural network.

There's a straightforward process in simple neural networks (let's compare them to a simple single-layered cake): input goes in, gets transformed, and output comes out. It's akin to having just one layer of sponge in our cake – simple.

Now, "deep learning" is like our multi-layered wedding cake. Instead of one layer, we have many, many layers stacked on top of each other. Each layer contributes to the overall flavor and structure of the cake. In neural networks, having multiple layers allows for more complex transformations of the input data, enabling the network to understand intricate patterns and details.

Why is depth important? Well, just as each layer of our cake can have a different flavor or texture, adding depth (or layers) to our network allows it to capture distinctive features and nuances in the data. For instance, in image recognition, initial layers might identify simple patterns like edges or colors, while deeper layers could recognize more complex features like shapes, and even deeper layers might identify objects or scenes. There are different hidden layer "recipes" for each kind of deep learning application (image recognition would use different stacks of layers than natural language processing, for example).

As with creating the most delicious cake, intuition and trial and error go into building the layers. A good baker can predict that a cake with alternating layers of raspberry and chocolate will turn out better than one with layers of lemon and chocolate. Similarly, a good AI engineer has some intuition as to what layer designs are more likely to work. But, just as with fusion cuisine, delightful surprises abound when new combinations are tried, and the reasons for the results are not always clear. The layers and how they are

combined are some of the hyperparameters that are pieces of the model (cake) you can set (vs. the parameters or weights learned in the training process).

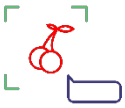
What can I do with Neural Networks?

At this point, you might be thinking, "Neat! But uh… *What* would I use neural networks for?" Below is a list of some of the more popular tasks for which neural networks are used. We have provided a small description of the underlying network types that make the task possible for each task. Beneath the list, there is a high-level description of the network types mentioned. This list is not intended to be exhaustive; you can do many more things with neural networks, with the limit mainly being your imagination and access to large, clean datasets. Network architectures are constantly changing as technology improves, so don't be confused if some of these are outdated by the time you take this course!

Computer visions tasks

- Computer vision is like giving eyes to a computer. It's the science of making machines "see" and interpret visual information from the world, like how humans use their eyesight.
- **Image classification**, **object detection**, and **segmentation** are some subtasks within computer vision.

1. **Image classification**
   - **What is it?** Classifying the primary object in an image into one of a predetermined set of categories. For example, the ImageNet dataset has 1,000 categories. Image classification algorithms predict the probability that an image belongs in each of those 1,000 categories. The highest probabilities are used to classify the image.



   - **Underlying Neural Network Technology:** Traditional image classification techniques used top-down, rules-based algorithms, but now, deep learning models, especially **convolutional neural networks (CNNs)**, dominate the scene. Initially designed for text, the transformer architecture is also making its way into computer vision, offering more flexibility and often better performance in certain tasks.
2. **Object detection**
   - **What is it?** Object detection involves identifying individual objects within an image, which is more complex than classifying an image as a whole. Multiple objects can be identified with an image, and each can be classified. Object detection typically involves placing a rectangular bounding box around individual objects within an image. Multiple objects can be identified with an image, and each can be classified. Object detection typically involves placing a rectangular bounding box around individual objects within an image.

   - **Underlying Neural Network Technology**: As with image classification, transformers are becoming increasingly popular in object detection. They're especially useful in tasks that require understanding the context and relationships between different parts of an image.
3. **Image Segmentation**

- **What is it?** Going a step beyond object detection, image segmentation identifies the pixels of objects in an image. As with all these methods, there are sub-categories with slightly different tasks and challenges. At the simplest level, the segmentation goes a step beyond object detection by labeling the pixels (vs a bounding box) of the detected objects (e.g., labeling the pixels of an image that are cat pixels and dog pixels). Image segmentation can be extended to instance segmentation, which labels each object, even of the same category, as distinct (e.g., dog1, dog2, etc.).

4. **Image Processing:**
    - **What is it?** Image processing involves manipulating or altering images to achieve a desired result. This can range from simple tasks like adjusting brightness and contrast to more complex ones like removing unwanted objects from a photo.
    - **Underlying Neural Network Technology**: Traditional image processing techniques used top-down, rules-based algorithms, but now, deep learning models, especially **CNNs**, dominate the scene. Initially designed for text, the transformer architecture is also making its way into image processing, offering more flexibility and often better performance in certain tasks.

Natural Language Processing (NLP):

- Natural Language Processing is the magic that allows computers to understand, interpret, and generate (!!!) human language. It's why chatbots can chat and translation apps can translate.
- Speech recognition, translation, and text generation are everyday tasks.

1. **Speech Recognition:**
    - **What is it?** Speech recognition is about converting spoken words into written text. It's the tech behind how voice assistants like Siri or Alexa "understand" what you're saying.

    - **Underlying Neural Network Technology**: Transformers' ability to capture long-range dependencies in data makes them well-suited for understanding the nuances and rhythms of spoken language.

2. **Translation:**
    - **What is it?** Translation involves converting text or speech from one language to another. Text translation used to use standard computer algorithms but has moved to deep learning models because they're more flexible and can "sense" context better than rule-based applications. It's like having a digital interpreter to translate sentences between languages instantly.

- **Underlying Neural Network Technology**: While earlier models used architectures like **Long Short-Term Memory (LSTMs)**, the current state-of-the-art for translation is the **transformer** architecture. Transformers have largely overshadowed other models due to their ability to handle long-range text dependencies and parallel processing capabilities. They've revolutionized machine translation, making it more accurate and efficient.

3. **Text generation**

- **What is it?** Tools like ChatGPT have recently become the face of AI in many ways. While they remain one small part of what AI is and how it is applied, the idea of a computer writing sentences, poems, computer code, and more has captured the imagination.
- **Underlying Neural Network Technology**: The Transformer architecture has revolutionized NLP. Models like Bidirectional Encoder Representations from Transformers (BERT), Generative Pre-trained Transformer (GPT), and their successors are all based on the transformer and have set new performance benchmarks in various language tasks. These models are trained on vast amounts of text, enabling them to understand context, nuances, and even sarcasm to some extent.

4. **Text to speech**

- **What is it?** From simply using prebuilt voice models to voice cloning technologies, text-to-speech tools convert text to spoken audio. You encounter this regularly with your smart assistants (Seri, Alexa, Google, etc.). Voice cloning and voice customization technologies are improving, and it is possible to train a model to generate audio that sounds like someone.
- **Underlying Neural Network Technology** As with much of NLP, transformers are responsible for huge advances in the field. Generative Adversarial Networks (GANs—see below) are used in voice cloning. Common methods convert input audio to an image, known as a spectrogram. The GAN is trained to generate spectrograms that look like the original. A different network, a Vocoder network, is used to convert the generated spectrograms to audio.

Time Series Analysis:

- **What is it?** A time series is like a chronological diary of numbers. Imagine noting your heartbeat rate every minute during a workout; that list forms a time series. Stock tickers are also time series data. Time series analysis detects patterns in the series to highlight anomalies or forecast trends.
- **Underlying Neural Network Technology**: While **Recurrent Neural Networks (RNNs)** and **LSTMs** were traditionally used for time series, transformers are now being adapted for this purpose. Their ability to capture patterns over long sequences makes them a promising tool for predicting future values in time series data.

Generative methods:

1. **Image Generation:**
   - **What is it?** This is about crafting new images from scratch. Tools that design pictures of fantasy creatures that have never been seen before are examples of Image Generators.

   - **Underlying Neural Network Technology**: While **Generative Adversarial Networks (GANs)** were the go-to for image generation, newer methods like **Stable Diffusion** have taken center stage. Stable Diffusion is a process that evolves an image over time, starting from noise and gradually refining it to produce a clear picture. It's like watching a foggy scene slowly come into focus.
2. **Image Translation:**
   - **What is it?** Image translation is the digital art of morphing one image into another. Think of turning a pencil sketch into a vibrant digital art piece.

   - **Underlying Neural Network Technology**: While **Conditional Generative Adversarial Networks (cGANs)** were popular for image translation, the combination of Transformers and Stable Diffusion techniques is now gaining prominence. This combo allows for more detailed and nuanced translations of images, capturing intricate details and styles.

Types of Networks

Here are some brief descriptions of the network technologies referenced above:

1. **Convolutional Neural Networks (CNNs):**

   CNNs are specially designed for processing grid-like data such as images.

   - **Convolutional Layer**: The cornerstone of CNNs. It uses filters (small, usually three-by-three, weight matrices) that slide over the input data (like an image) to produce a feature map. This operation helps detect local features like edges, textures, or patterns.
   - **Pooling Layer**: Typically used after convolutional layers, pooling layers reduce the spatial dimensions of the feature maps, keeping the essential information. A standard method is max pooling, which retains the maximum value from a section of the feature map.
   - **Fully Connected Layer**: After several convolutional and pooling layers, high-level reasoning about the features occurs in fully connected layers. They connect every neuron in one layer to every neuron in the next layer, much like traditional neural networks.

2. **Recurrent Neural Networks (RNNs):**

   RNNs are designed to process sequential data by keeping a "memory" of earlier inputs in their internal state.

   - **Memory:** Unlike traditional neural networks, an RNN retains a form of memory. It achieves this by keeping a hidden state that gets updated at each time step of a sequence.

- **Sequence Processing:** Given a data sequence (like a sentence broken down into individual words or a time series), an RNN processes one element at a time while updating its hidden state. This allows it to carry forward information from earlier parts in the sequence.
- **Vanishing Gradient Problem:** A significant challenge with basic RNNs is the vanishing (or exploding) gradient problem. This means that during training, the gradients—which are used to update the network's weights—can become extremely small or large, making it hard for the RNN to learn long-term dependencies.

3. **Long Short-Term Memory (LSTM):**

   LSTMs are a special kind of RNN designed to address the vanishing gradient problem and better capture long-term dependencies in data.

   - **Cell State:** The core idea behind LSTMs is the cell state, a kind of "conveyor belt" that runs through the LSTM unit. Information can be added to or removed from this cell state via gates.
   - **Gates:** LSTMs use three gates:
     - **Forget Gate:** Decides what information from the cell state should be thrown away or kept.
     - **Input Gate:** Updates the cell state with new information.
     - **Output Gate:** Decides what information based on the cell state should be output from the LSTM unit.

   These gates enable LSTMs to selectively remember or forget information, making them better suited for tasks that require understanding over longer sequences.

4. **Transformers:**

   Transformers revolutionized the NLP field and are now finding applications in other domains.

   - **Attention Mechanism**: At the heart of the transformer is the attention mechanism. It allows the model to focus on different parts of the input data with varying (numerical) intensity. The "self-attention" mechanism lets each word in an input sequence look at every other word to determine its context.
   - **Positional Encoding**: Since transformers don't have a built-in sense of sequence (like RNNs do), positional encoding is added to give the model information about the position of words in a sequence.
   - **Stacked Layers**: A transformer holds multiple stacked layers of these attention mechanisms, allowing for complex patterns and relationships in the data to be captured.

5. **Generative Adversarial Networks (GANs):**

   GANs are designed for generative tasks, like creating images or other forms of data.

   - **Two-Part Structure:** A GAN consists of two neural networks—a generator and a discriminator—trained simultaneously through adversarial training.
   - **Generator:** This network takes random noise as input and generates data (like an image).

- **Discriminator:** This network tries to distinguish between genuine data samples and fake samples produced by the generator.
- **Adversarial Training:** During training, the generator tries to produce data that the discriminator can't distinguish from real data. In contrast, the discriminator tries to distinguish real data from fake better. It's like a cat-and-mouse game where both parties evolve together.

The goal is to have a generator so good at its task that the discriminator can't tell the difference between the real and generated data.

6. **Conditional Generative Adversarial Networks (cGANs):**

   cGANs are an extension of traditional GANs, tailored for controlled data generation based on specific conditions.

   - **Conditioned Generation:** Unlike standard GANs, cGANs incorporate a conditional input to guide the data generation process. This condition could be a label, an image, or any relevant information.
   - **Modified Structure:**
     - **Generator:** Receives random noise and conditional input, aiming to produce data that aligns with the provided condition.
     - **Discriminator:** Evaluates the authenticity of data samples, considering both the sample and its associated condition.
7. **Stable Diffusion:**

   While GANs utilize a generator-discriminator setup for data generation, Stable Diffusion models take a different approach based on diffusing data.

   - **Diffusion Process**: The idea is to view data generation as a reverse diffusion process. Given a data point (like an image), it's gradually corrupted with noise over a series of steps until it's transformed into pure noise.
   - **Reverse Process**: The generative task involves reversing this process. Starting with noise, the model aims to remove the noise step-by-step, reconstructing the original data point.
   - **Stability**: One of the advantages of Stable Diffusion over GANs is its stable training process. Whereas GANs can suffer from issues like mode collapse (where the generator produces limited varieties of samples), Stable Diffusion models provide a more stable and consistent way to generate diverse, high-quality data samples.

Again, these lists are subject to the rapid advancement of AI techniques and architectures. If you're interested in the latest model designs or the kinds of tasks being performed with AI, a **model zoo** would be the place to look. Model zoos are repositories of pre-trained machine learning models—these are like cookbooks. Find and use a good recipe or adapt it to your needs! Instead of building and training a model from scratch, which can be time-consuming and resource-intensive, researchers and developers (like you!) can visit a model zoo to find a model that suits their needs. These pre-trained models can then be used directly or fine-tuned for specific tasks, offering a head start in various applications. Model zoos act as a bridge, facilitating the sharing and dissemination of knowledge within the machine-learning

community. One such zoo is Hugging Face, where hundreds of thousands of pre-trained models are organized by tasks: Huggingface Tasks. We encourage you to check it out!

Python AI Frameworks:

As we have seen, libraries like Pandas and NumPy make using Python for specific tasks more manageable. As AI applications were being developed, coders built frameworks (which can be considered extensive collections of libraries that enable complex tasks—like doing AI research) to facilitate their AI research. Competing teams of coders, mostly at large companies like Meta (Facebook) and Alphabet (Google), developed competing frameworks. While the primary goals and many underlying methods are similar, these frameworks have different ways of interacting with them and different strengths and weaknesses. The most common frameworks are PyTorch and TensorFlow. Keras started as an easy-to-use framework for interacting with other frameworks. At one time, it focused on TensorFlow, but as of version 3, released in Fall 2023, it now supports TensorFlow, PyTorch, and JAX.

Which framework to use?

This is a challenging question to answer. At *Practicum AI*, we have moved to the PyTorch Lightning framework. PyTorch Lightning uses a modular and organized code structure that makes it more readable for users that are new to Python. For experienced users, it facilitates scaling with minimal code. The big takeaway is that we find it's easier to teach and learn than other frameworks.

Another popular framework is Tensorflow with Keras. Tensorflow is considered to have one of the broadest ecosystems of tools for things like data preparation and model deployment. While Tensorflow with Keras is incredibly powerful for creating deep learning models, the code tends to be less abstracted than PyTorch Lightning and thus harder to follow in a lesson.

Ultimately, it is worth looking at the papers published in your field of interest and seeing what the authors use. It may be easier to advance your field by sticking with the same framework others are using. It's also important to note how fast this frameworks are advancing. When we wrote the first version of this course in 2023 (😅), Tensorflow with Keras was champion of readability in deep learning! Staying on top of framework development is at least as important as keeping track of important architectures and techniques in your field.

Lastly, these are all frameworks to implement aspects of AI model training. Once you understand what you are doing, it is not difficult to use the strengths of different frameworks to your advantage and mix and match methods.

Image recognition Exercise

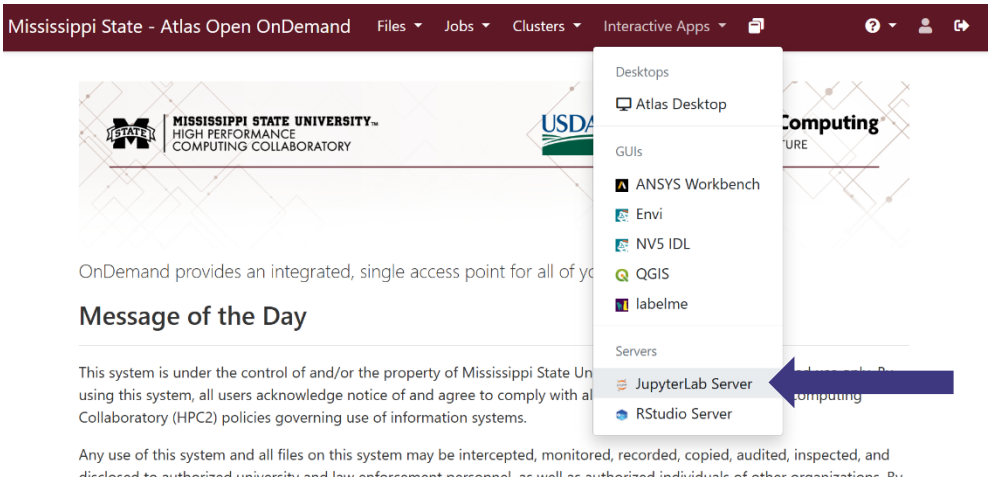
Now that we've thoroughly explored the basics of neural networks, let's get our hands dirty with creating one ourselves. The following exercise will use the Python coding language inside a Jupyter Notebook environment to give you a quick introduction to doing AI with code!

The notebooks for the Deep Learning Foundations course are located at

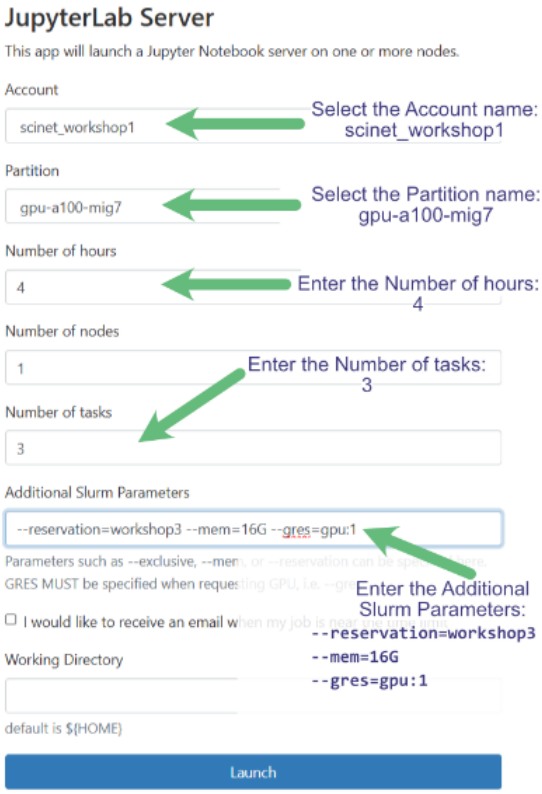
- https://github.com/PracticumAI/deep_learning_pt-lightning_ars
- As in the past, click the Use this template button to make your copy of the repository. Start an Atlas session as directed below (we will want a GPU for these exercises).

Launch a Jupyter session on Atlas

- Login at: https://atlas-ood.hpc.msstate.edu/
- For this course, we want to start requesting a GPU. This is done using the JupyterLab Server option from the Interactive Apps menu:



- The resource selections are a bit different than those used in previous courses. Use these settings; see the image for details.
  - Account Name: scinet_workshop1
  - Partition Name: gpu-a100-mig7
  - Number of hours: 4
  - Number of nodes: 1
  - Number of tasks: 3
  - Additional Slurm Parameters:
    `--reservation=workshop3`
    `--mem=16G --gres=pgu:1`



Deep Learning Foundations

- Once your session starts, connect to Jupyter
- Clone your repository to Atlas. (`git clone <paste the SSH link from GitHub>`)
- To set up the Python environment with all the modules required for these exercises, open the notebook **00_kernel_setup.ipynb** on Atlas and run the commands noted in a terminal.
- In the Launcher tab of Jupyter, there should now be a "dlf_workshop" kernel card. This is the kernel you will want to use for the notebooks in this course.

- When you open a notebook, they should all default to the "dlf_workshop" kernel, but if needed, click on the kernel name in the top right of the notebook and select "dlf_workshop" from the dropdown list.
- Work through notebook **01_deep_learning_tour.ipynb**