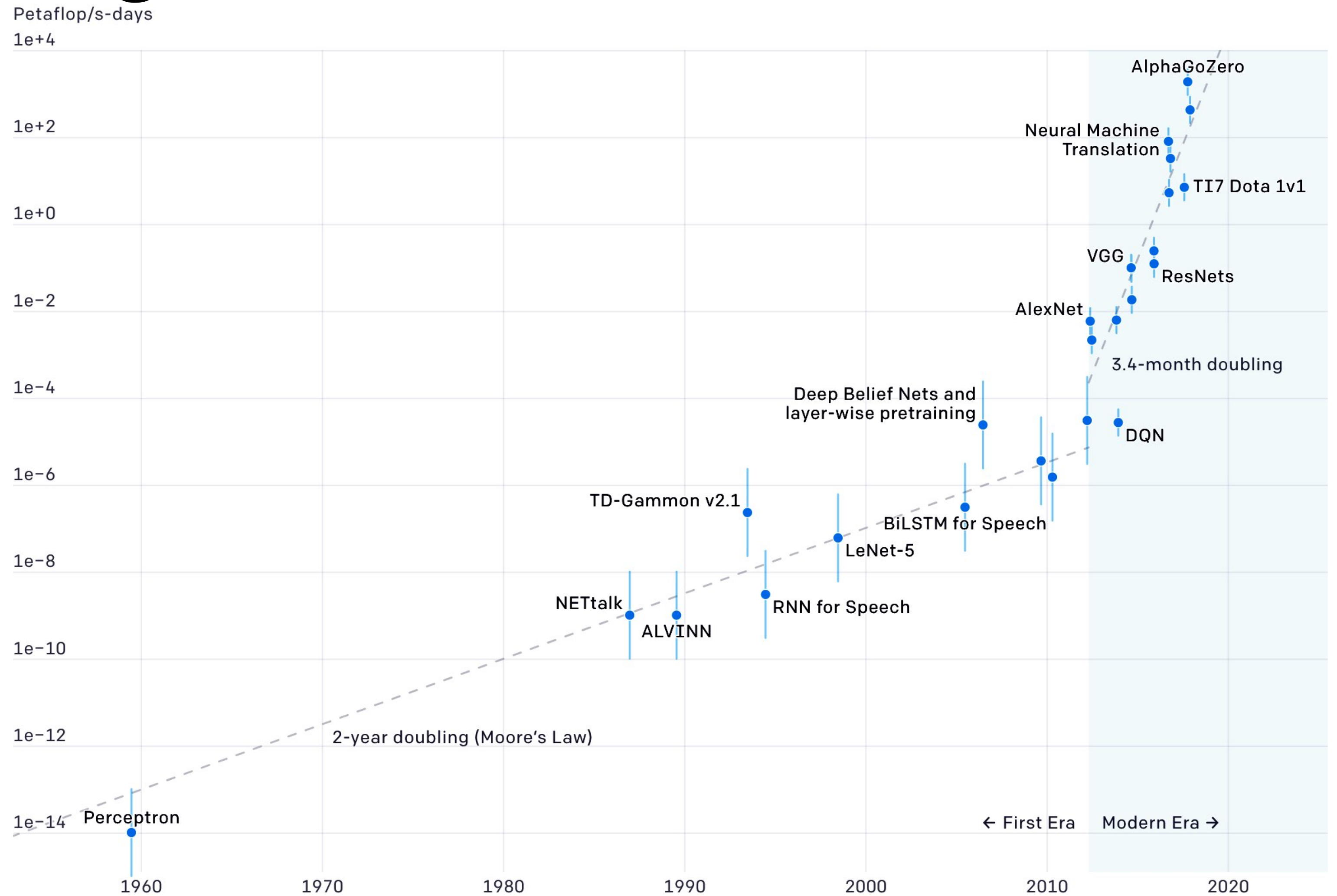


# **Deep Learning with GPU cores**

**How to do more in less time**

# Why Deep Learning + GPUs?

- Deep learning is bound by compute power.
- GPU enable efficient training of neural networks.



# What we'll do

	Deep Learning with GPU cores	
09.30 - 09.45	Welcome	
09.45 - 10.15 <b>(30 min)</b>	Deep Learning and Infrastructure	Learn how to train a neural network with a GPU.
10.15 - 11.30 <b>(60 min)</b>	Practical: Working on the GPU	
11.30 - 11.45	Short break ☕	
11.45 - 12.00 <b>(15 min)</b>	Introduction to Profiling	Learn how to profile the training and training efficiently.
12.00 - 12.45 <b>(45 min)</b>	Practical: Profiling Jobs	
12.45 - 13.00	General Q&A	

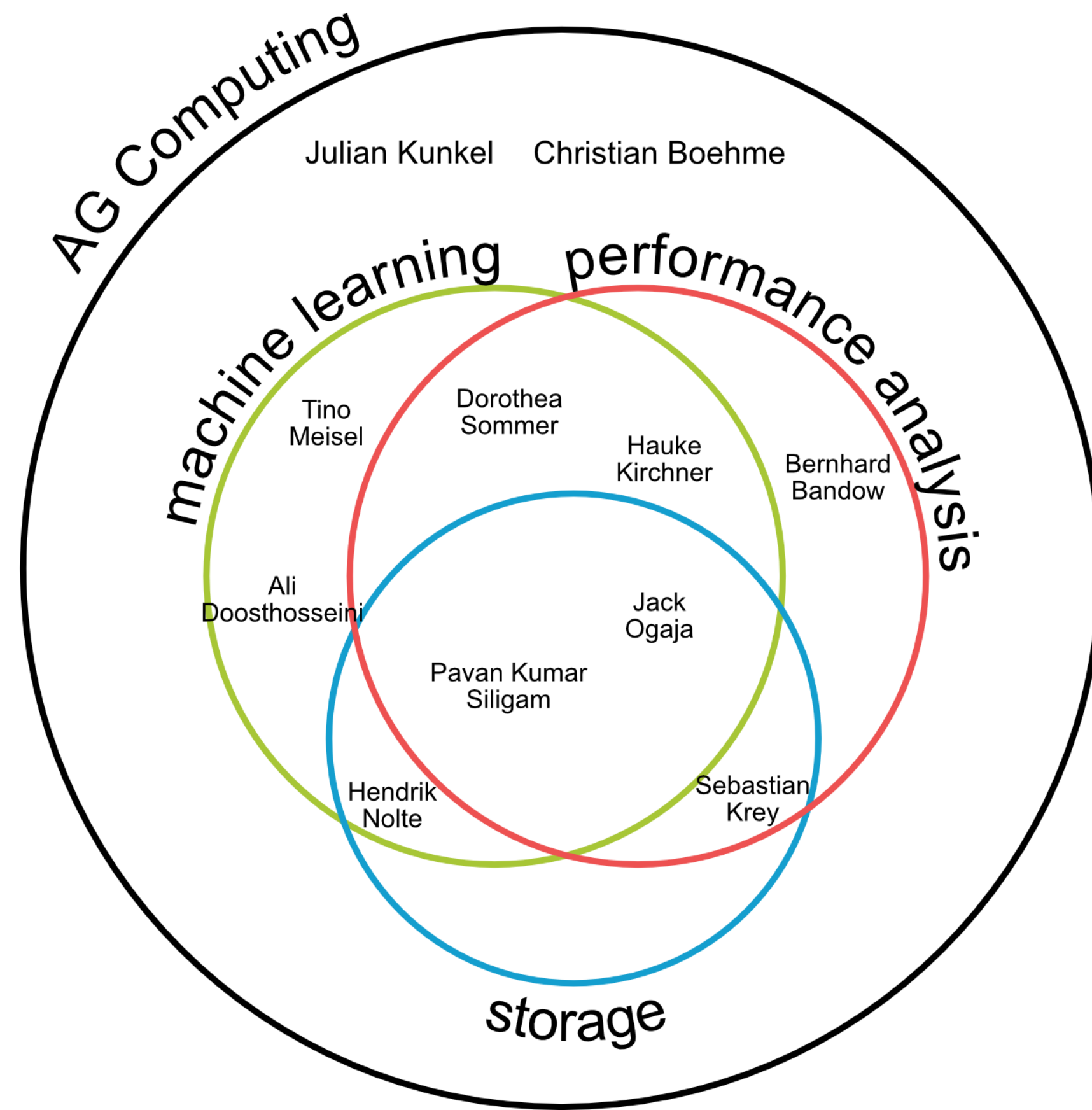
# Why do we offer this course?



- Working Group “Computing” at the GWDG
- Work in conjunction with University Göttingen and Max Planck Society
- **Mission:** provide scalable solutions for resource-intensive applications
  - Independent research in the field of computer science
  - Planning, operation, hosting and housing of HPC systems
  - Training around the use of HPC systems



# Working Group Computing



We are currently 40 persons.  
You can become part of our team!

We are  
... supervising theses.  
... hiring.

# Dorothea Sommer

- Data Scientist at AG Computing
- Experience
  - M.Sc. Computational Neuroscience
  - Meta-Learning for Reinforcement Learning (Master Thesis)
  - Normative Modeling for Computer Vision (Charité)
- Research Interest
  - Machine Learning
  - Forest science
  - Research Software



# Hauke Kirchner

- Data Scientist at AG Computing
- Experience
  - B.Sc. Biology (Göttingen)
  - M.Sc. Forest Information Technology (Eberswalde, Warschau)
  - Tree species classification from airborne LiDAR using individual crowndelineation and machine learning (Master Thesis, UFZ)
- Research Interest
  - Machine Learning
  - Remote sensing
  - Forest science
  - Data management





# Tino Meisel

- Data Scientist at AG Computing
- Experience
  - M. Sc. Physics
  - Certified Data Scientist
  - Time Series Prediction and Classification of COVID-19 data
  - Feature Recognition in AFM, TEM
- Research Interest
  - Deep Learning
  - Scientific Machine Learning
  - Quantum Computing (QML)



# Who are you?

<https://take.supersurvey.com/QS7GQLNYR>



# What we'll do

	Deep Learning with GPU cores	
09.30 - 09.45	Welcome	
09.45 - 10.15 <b>(30 min)</b>	Deep Learning and Infrastructure	Learn how to train a neural network with a GPU.
10.15 - 11.30 <b>(60 min)</b>	Practical: Working on the GPU	
11.30 - 11.45	Short break ☕	
11.45 - 12.00 <b>(15 min)</b>	Introduction to Profiling	Learn how to profile the training and training efficiently.
12.00 - 12.45 <b>(45 min)</b>	Practical: Profiling Jobs	
12.45 - 13.00	General Q&A	

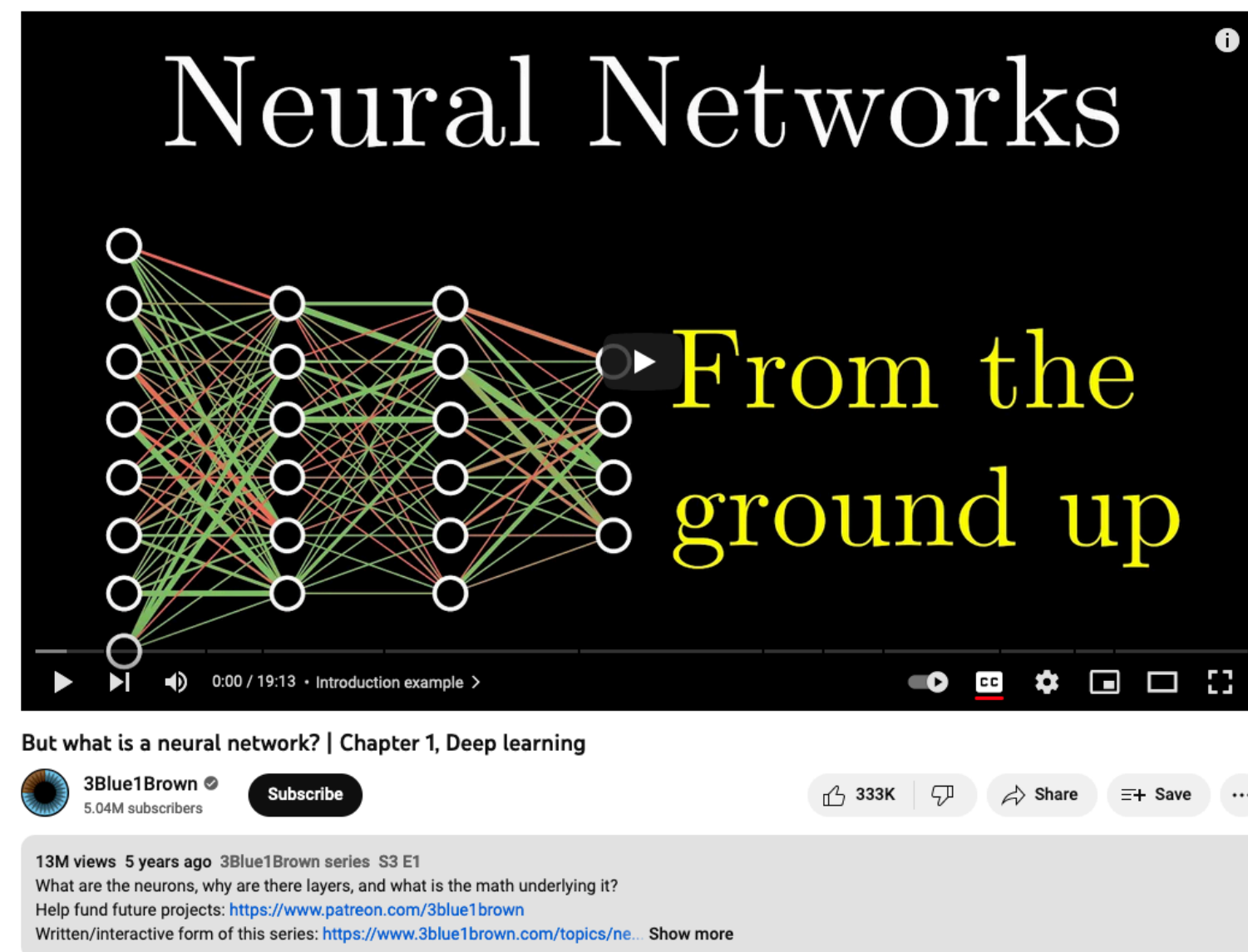
# **Deep Learning and Infrastructure**

**Deep Learning Example**

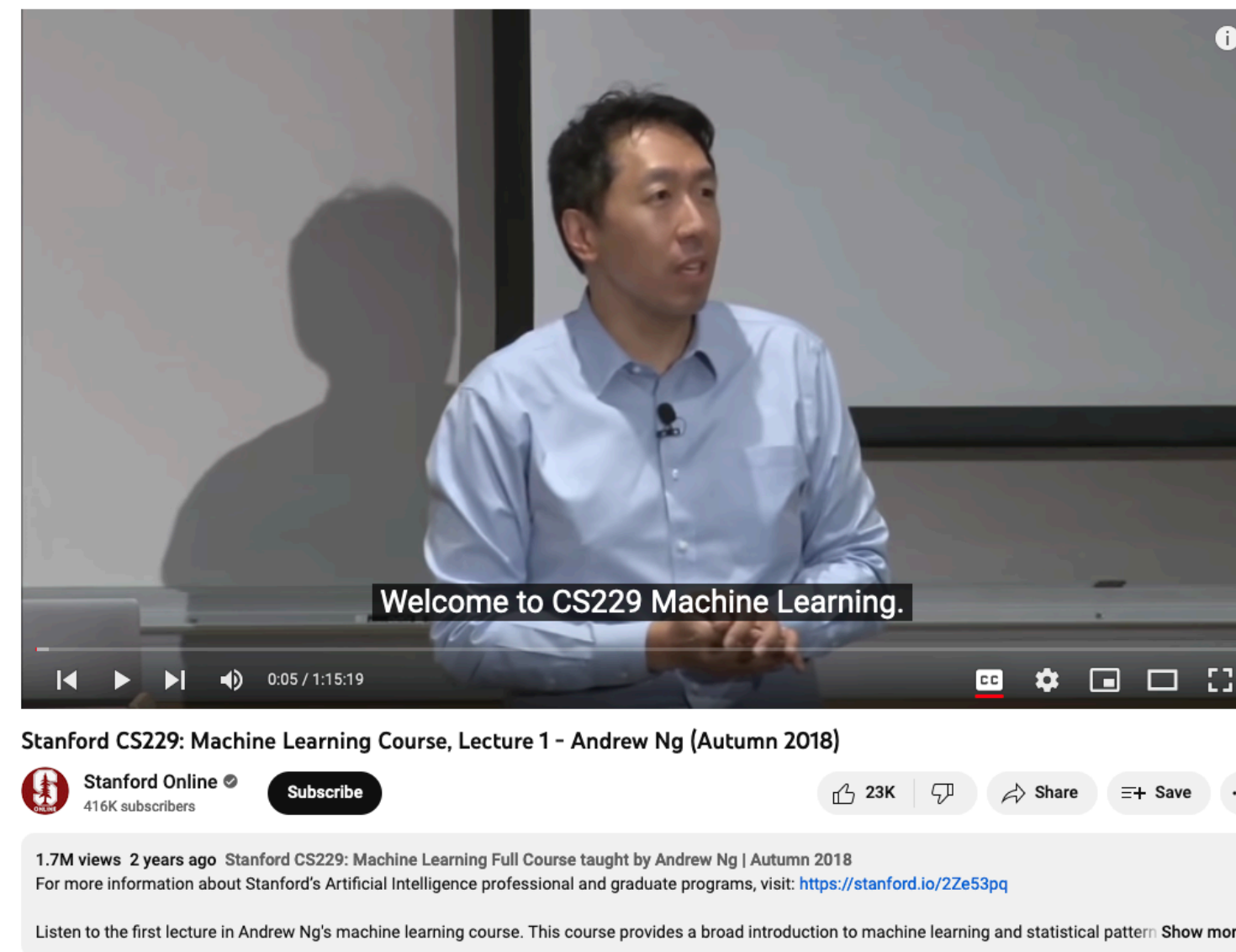


# Going further

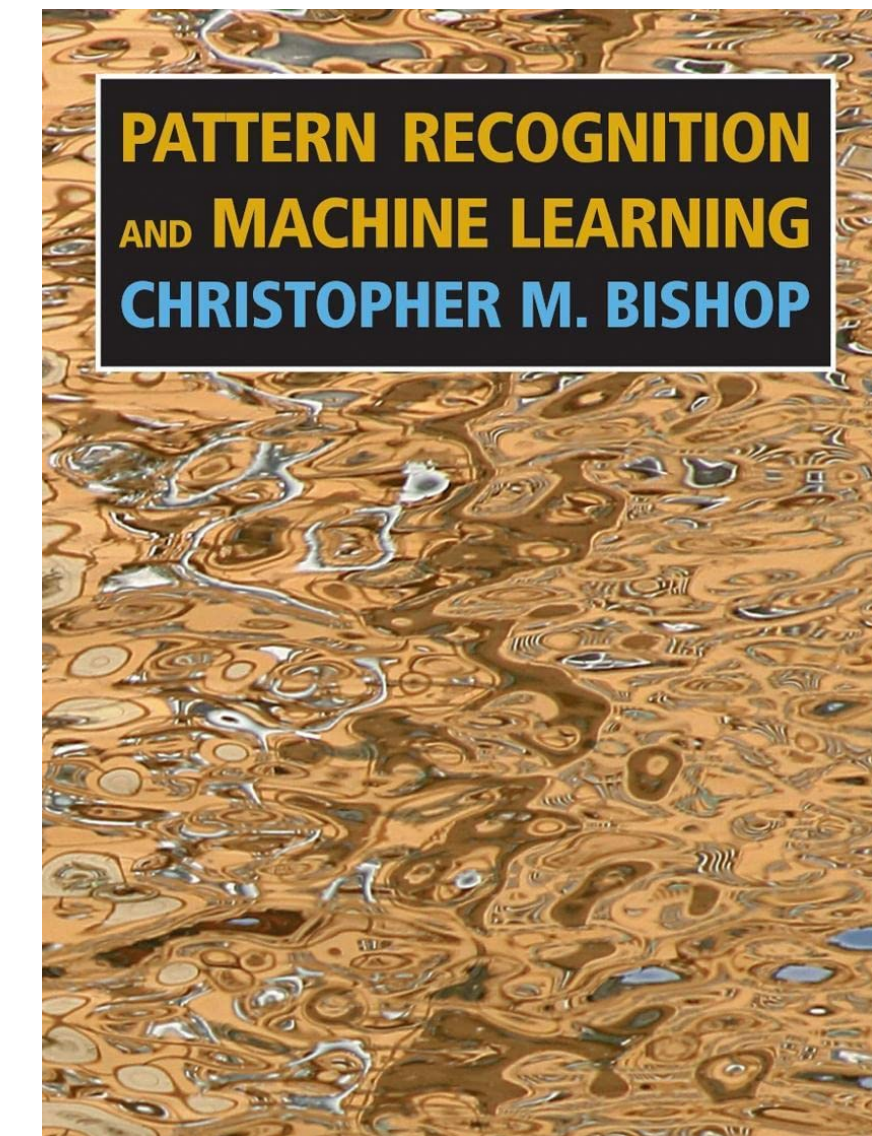
Requires more effort (time and math)



- YouTube series by **3Blue1Brown**:
- explains the math behind neural networks intuitively
  - amazing visualisations



- YouTube series by **Andrew Ng**:
- clear explanation of concepts
  - involves basic math
  - use older series if you like more math



- Classic book by **Christopher Bishop**:
- good if you have a solid math background



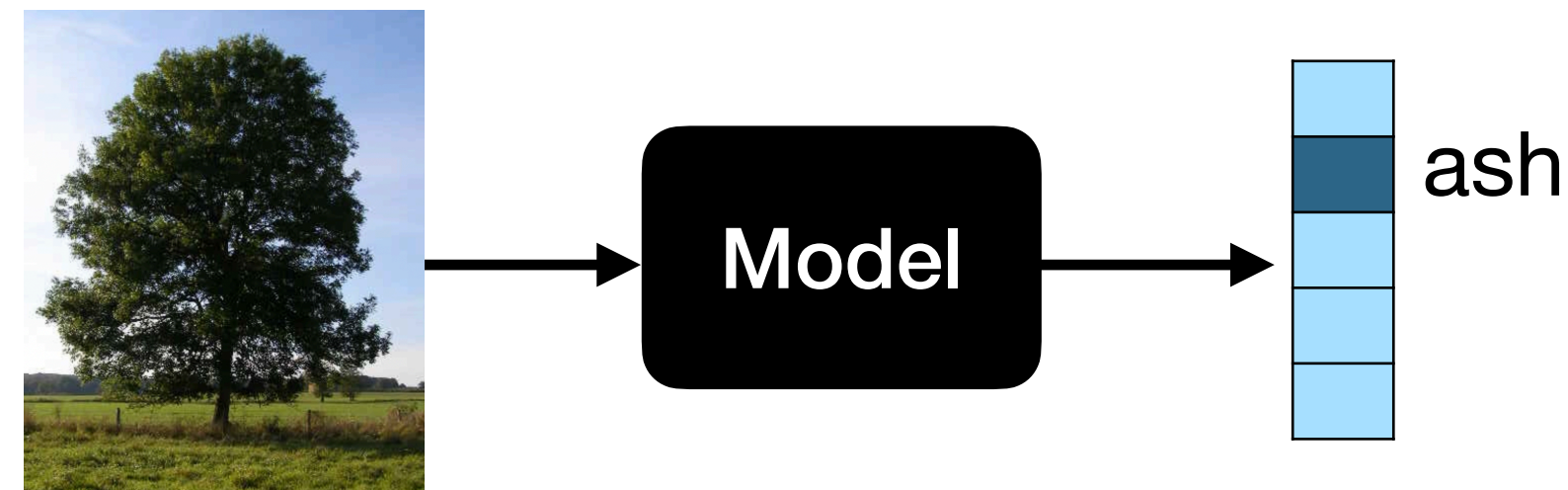
# Learning Paradigm

## Use Case

*tree species classification*

$\underline{x}_i$  tree photo

$y_i$  corresponding tree species



### classification

predict tree species  
given photo of the tree

## Supervised Learning

*“learning with teacher”*

Observations  $\underline{x}_1, \dots, \underline{x}_n$

Labels  $y_1, \dots, y_n$



### classification

predict label of observation,  
learning with ground truth

## Let's build...



3.3★  
19.5K reviews ⓘ

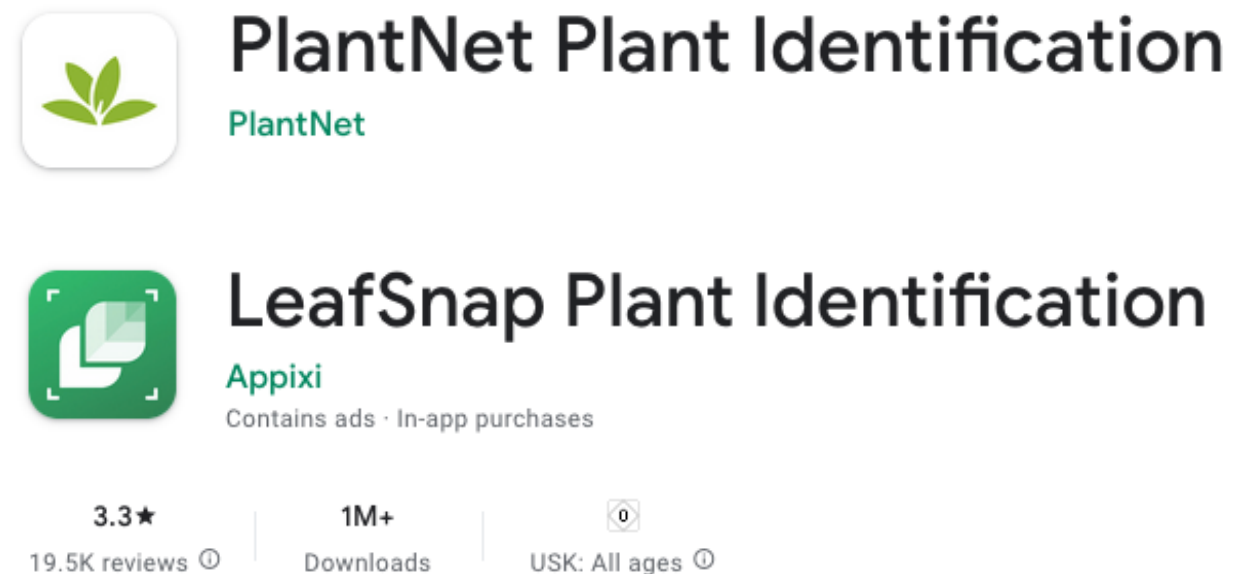
1M+  
Downloads

USK: All ages ⓘ

**Snap picture of tree  
to identify species**

# Deep Learning

## Let's build...

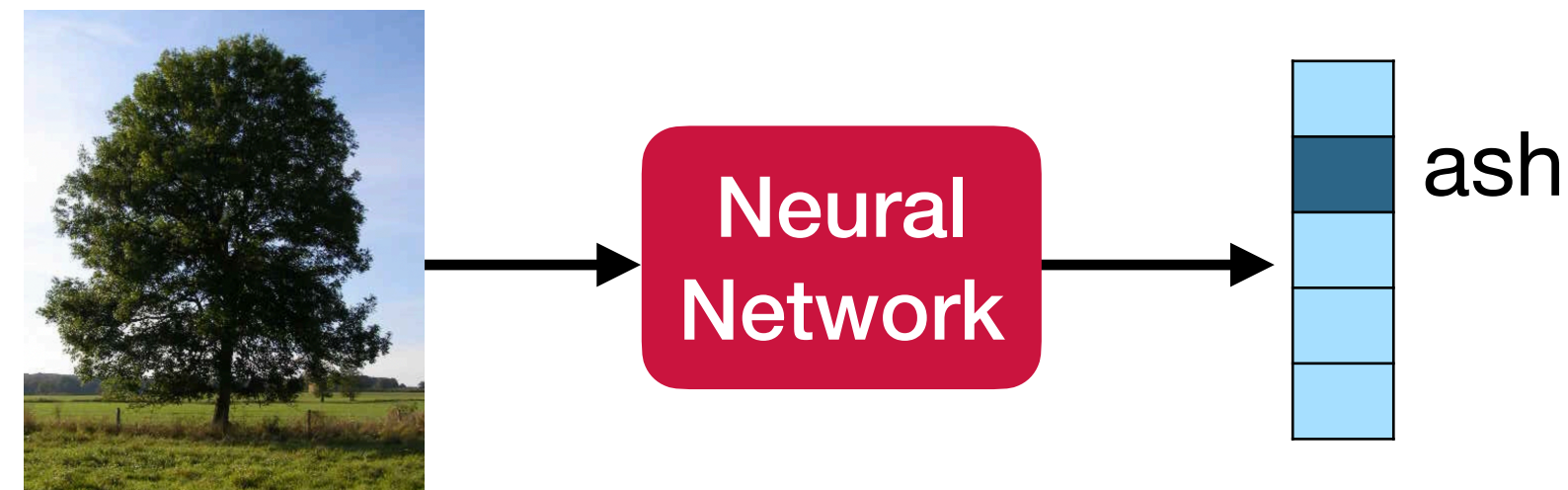


**Snap picture of tree  
to identify species**

## Use Case

*tree species classification*

$\underline{x}_i$  tree photo  
 $y_i$  corresponding tree species

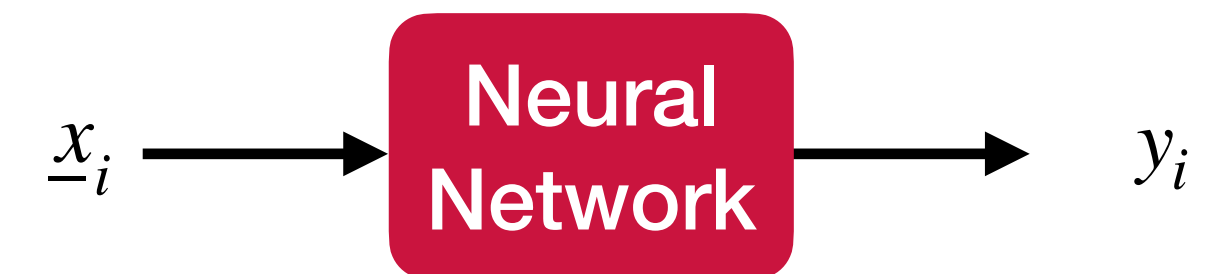


**classification**  
predict tree species  
given photo of the tree

## Supervised Learning

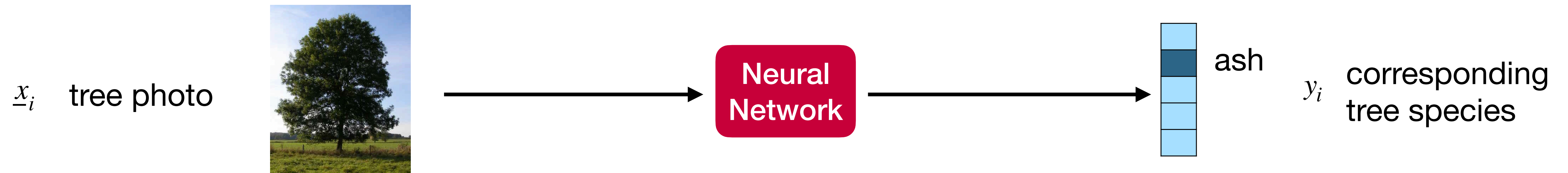
*“learning with teacher”*

Observations  $\underline{x}_1, \dots, \underline{x}_n$   
Labels  $y_1, \dots, y_n$

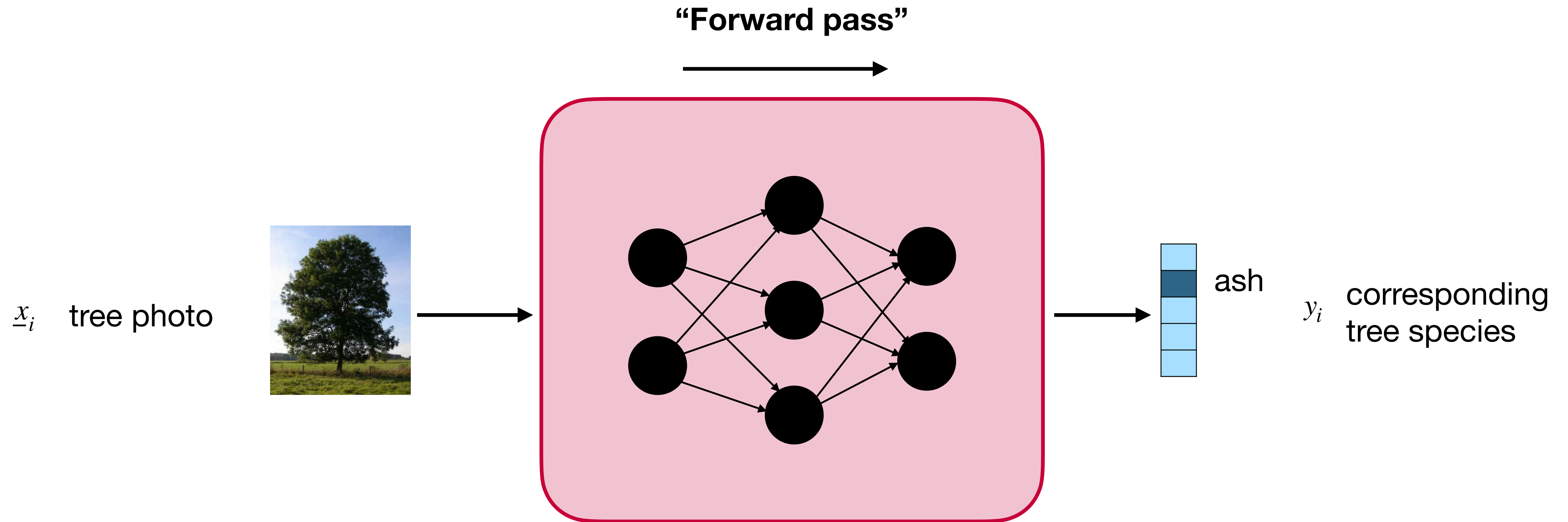


**classification**  
predict label of observation,  
learning with ground truth

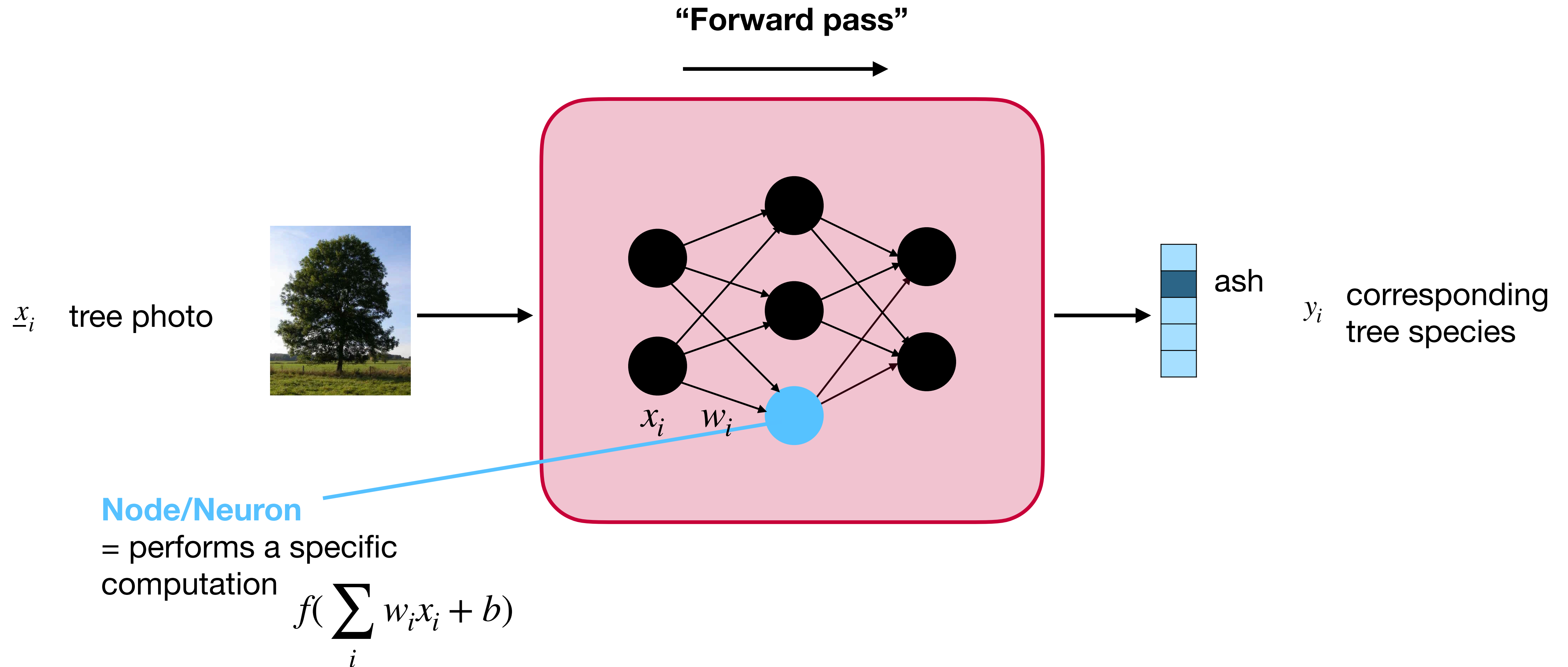
# What happens in the network?



# What happens in the network?

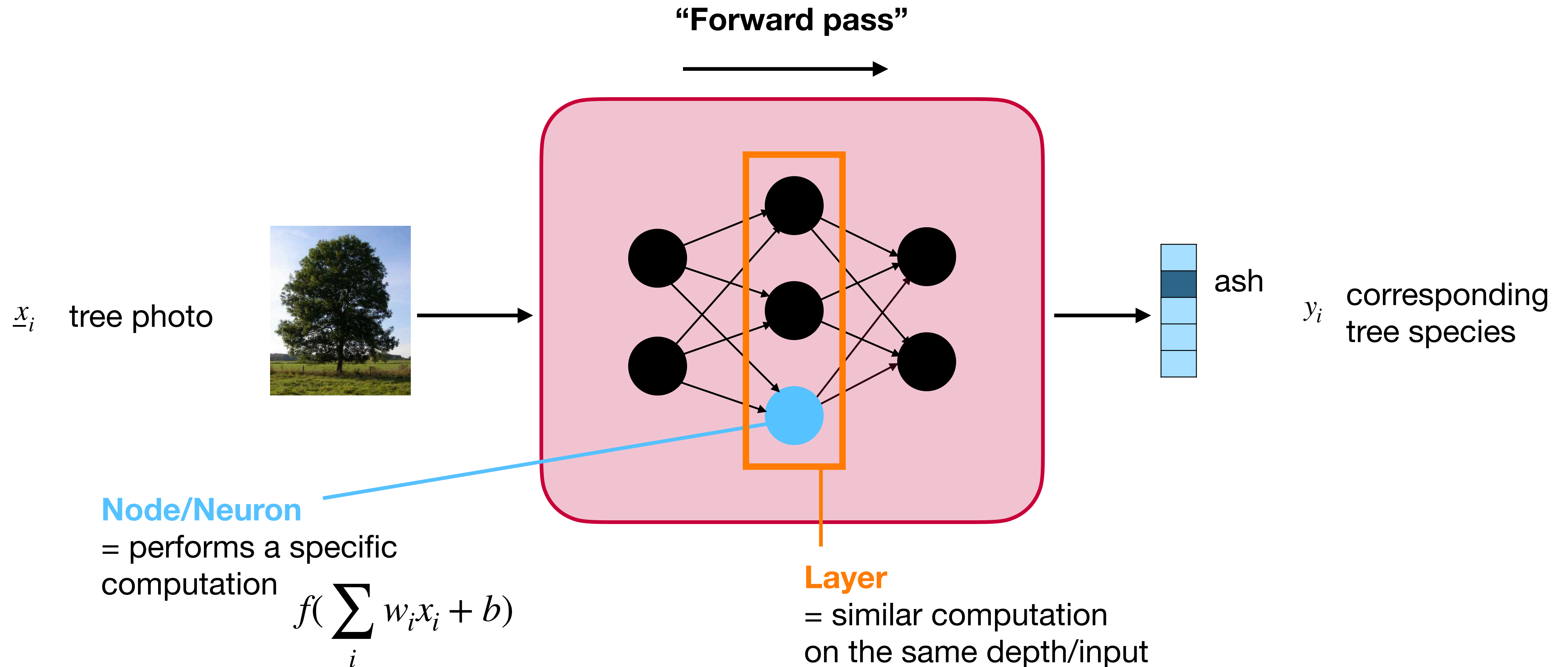


# What happens in the network?



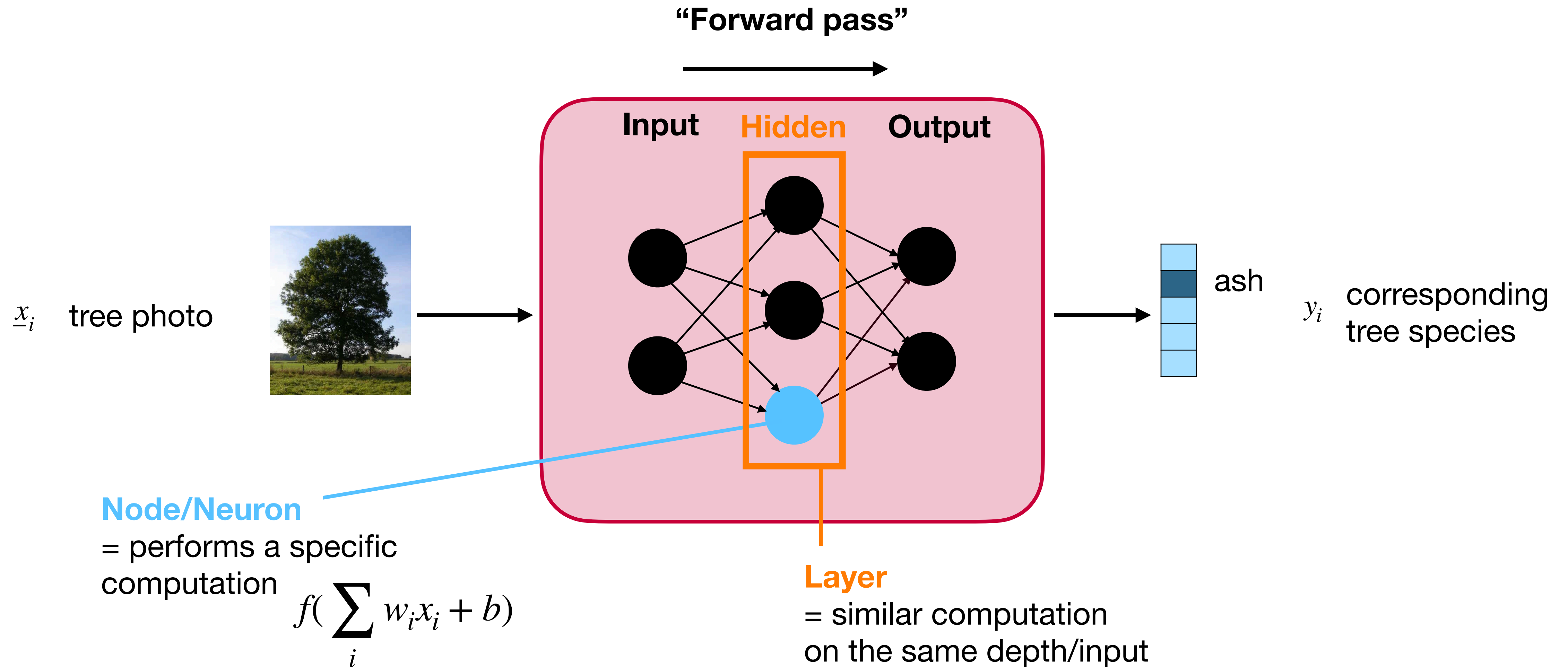


# What happens in the network?

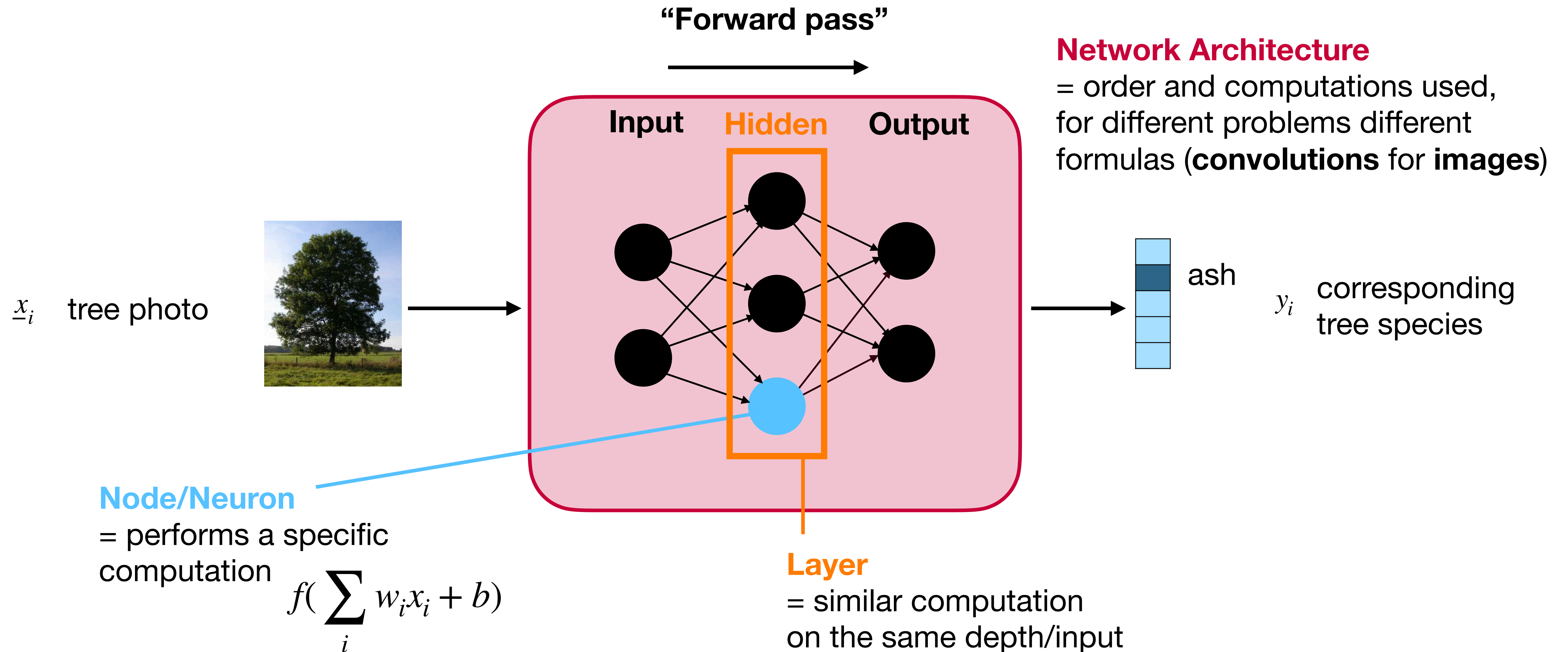




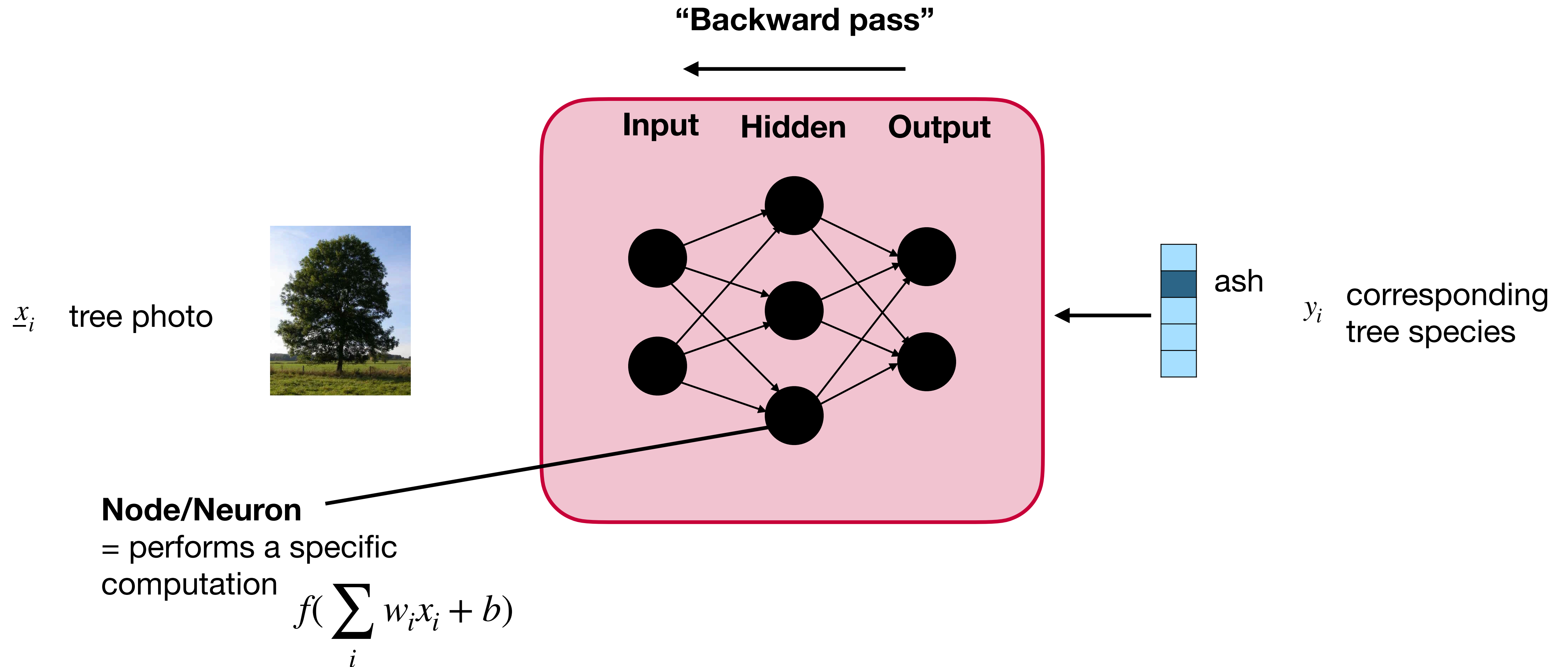
# What happens in the network?



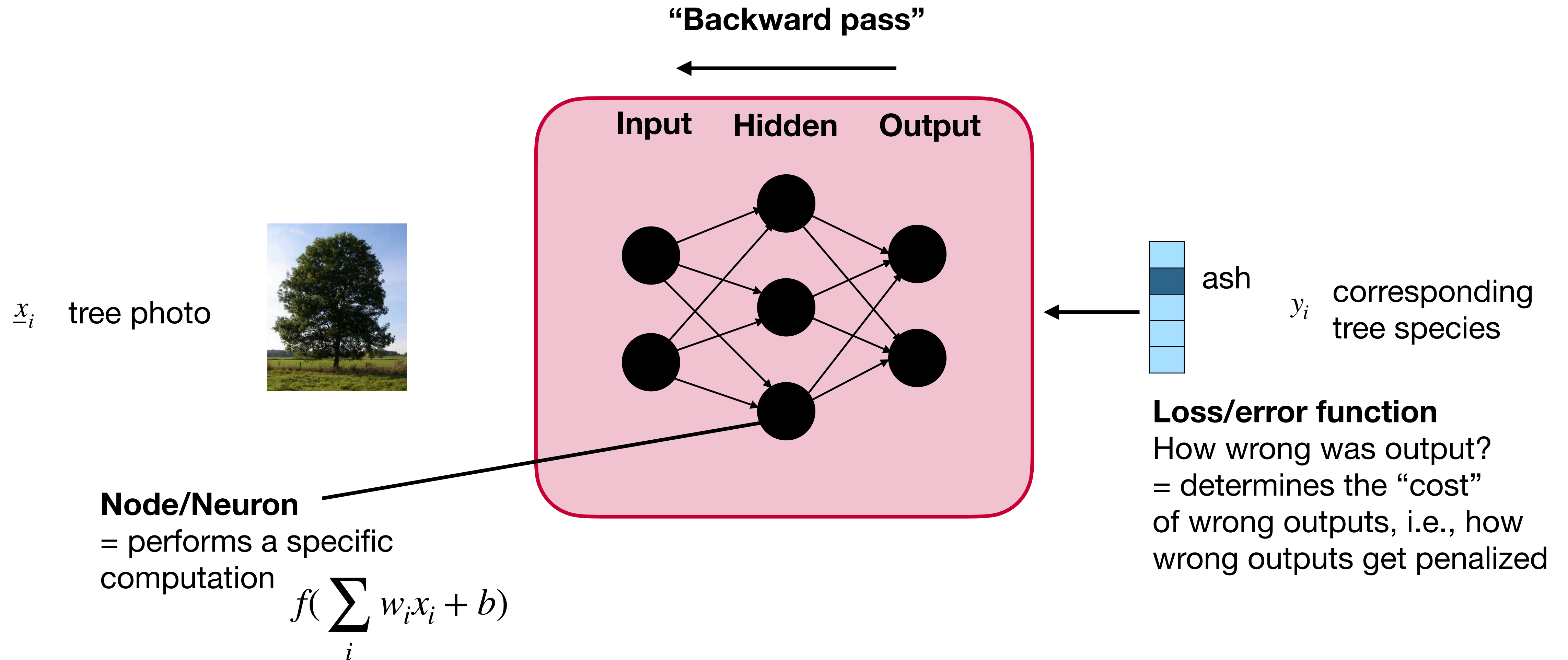
# What happens in the network?



# What happens in the network?

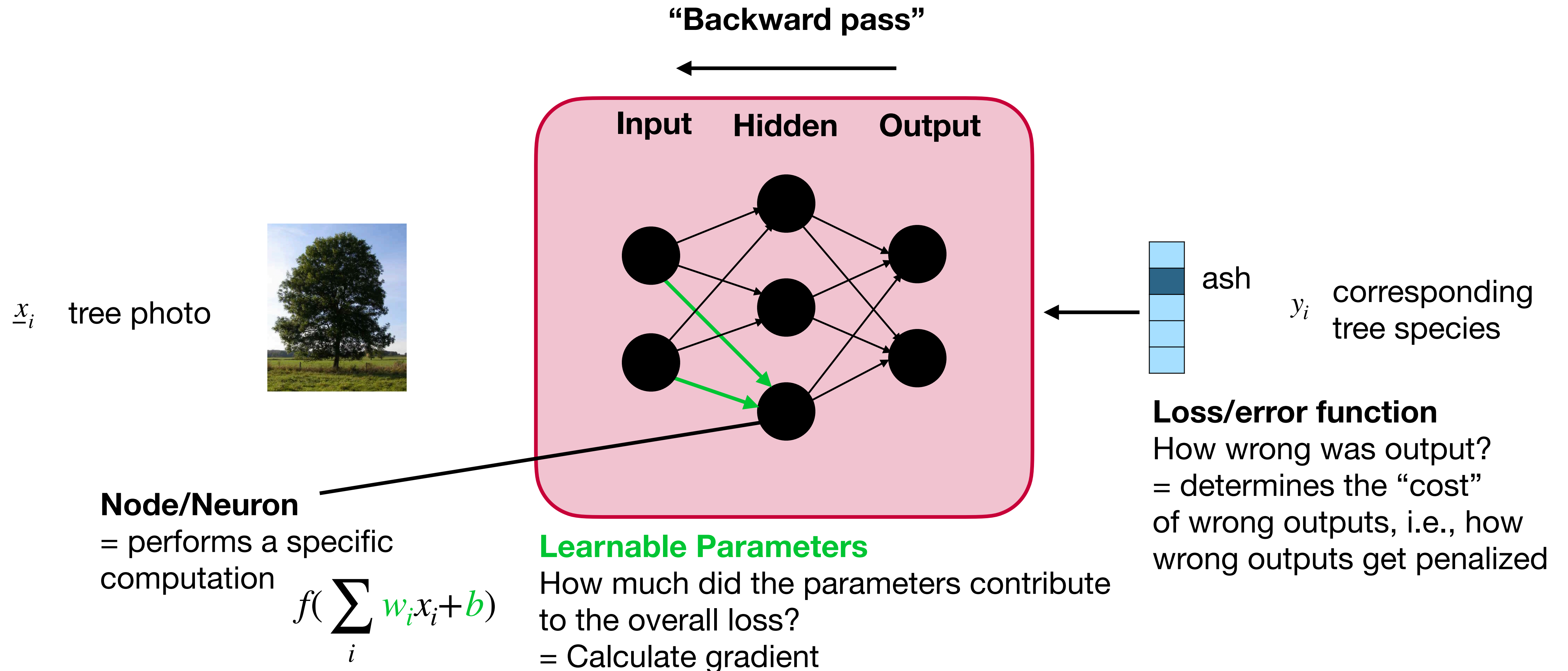


# What happens in the network?

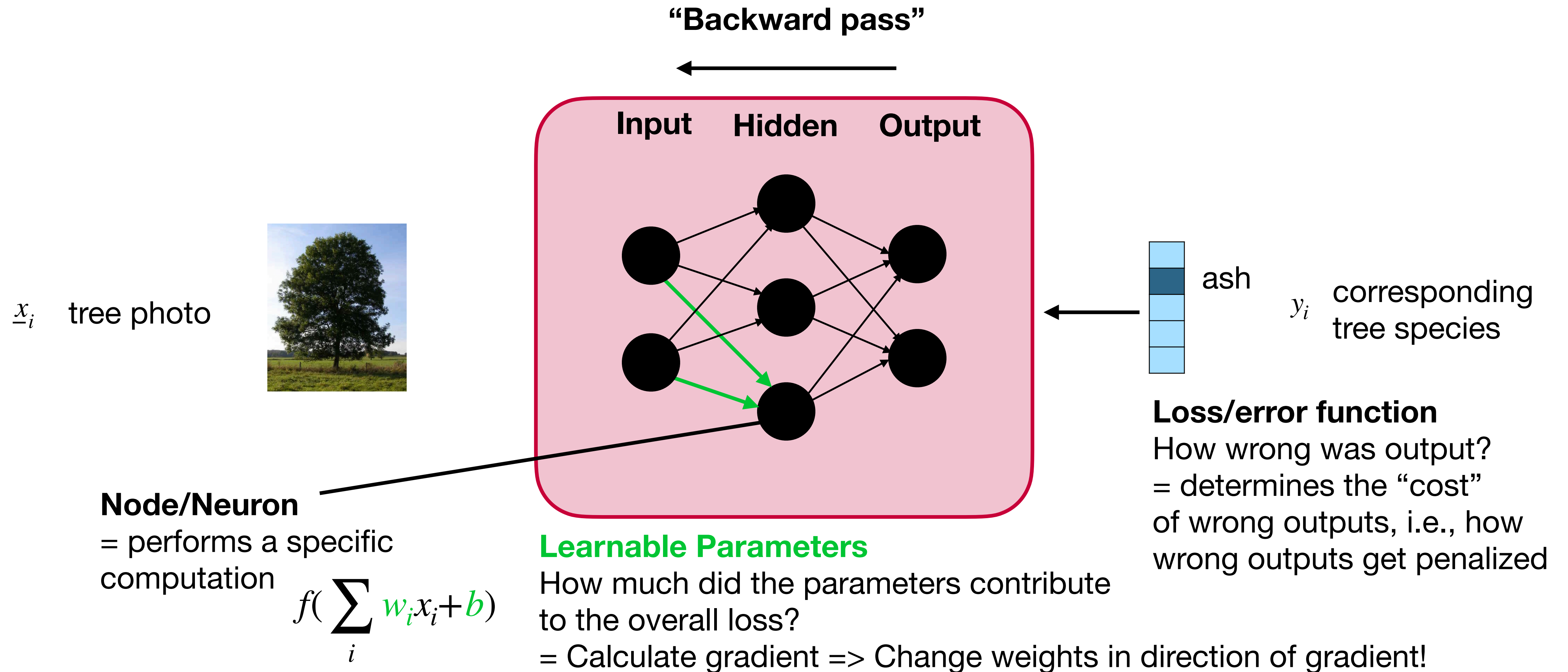




# What happens in the network?



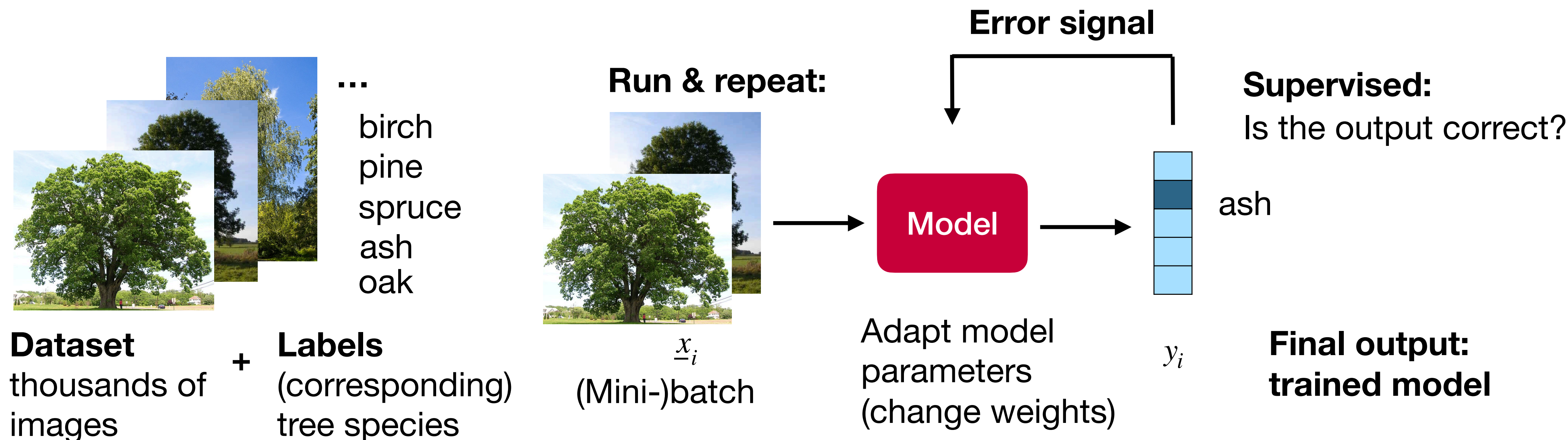
# What happens in the network?





# Deep Learning: Procedure

## Training

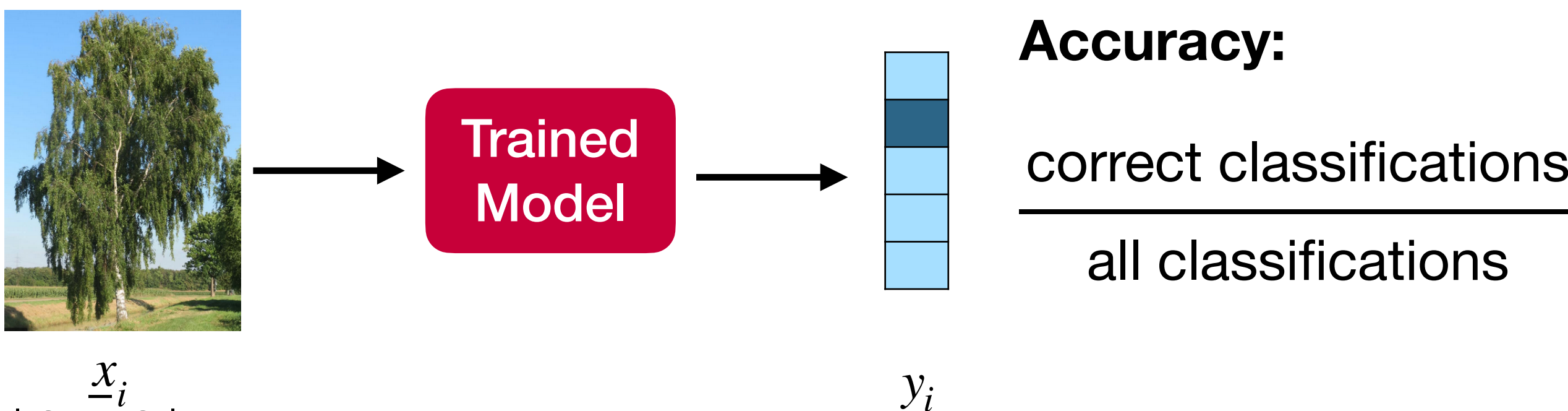


## Testing

How well does our model generalise?

**Dataset** of *unseen* images

### Inference:





# Where do we need GPUs?

## Training



**Dataset**  
thousands of  
images

+

**Labels**  
(corresponding)  
tree species

...  
birch  
pine  
spruce  
ash  
oak

**Run & repeat:**



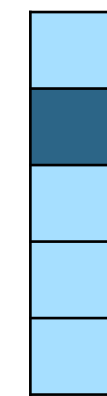
$\underline{x}_i$   
(Mini-)batch



**Model**



Adapt model  
parameters  
(change weights)



$y_i$

**Error signal**



**Supervised:**  
Is the output correct?

ash

**Final output:**  
**trained model**  
(model parameters)

**Inference:**

## Testing

**How well does  
our model  
generalise?**

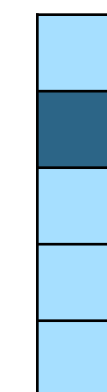
**Dataset**  
of *unseen*  
images



$\underline{x}_i$



**Trained  
Model**



$y_i$

**Accuracy:**  
$$\frac{\text{correct classifications}}{\text{all classifications}}$$

# **Deep Learning and Infrastructure**

## **Checklist**

# Training Checklist

## Tools & Infrastructure

- Place to train (access to cluster)
- Cluster essentials
- Data
- Code
- Monitoring & tracking



# Where to train

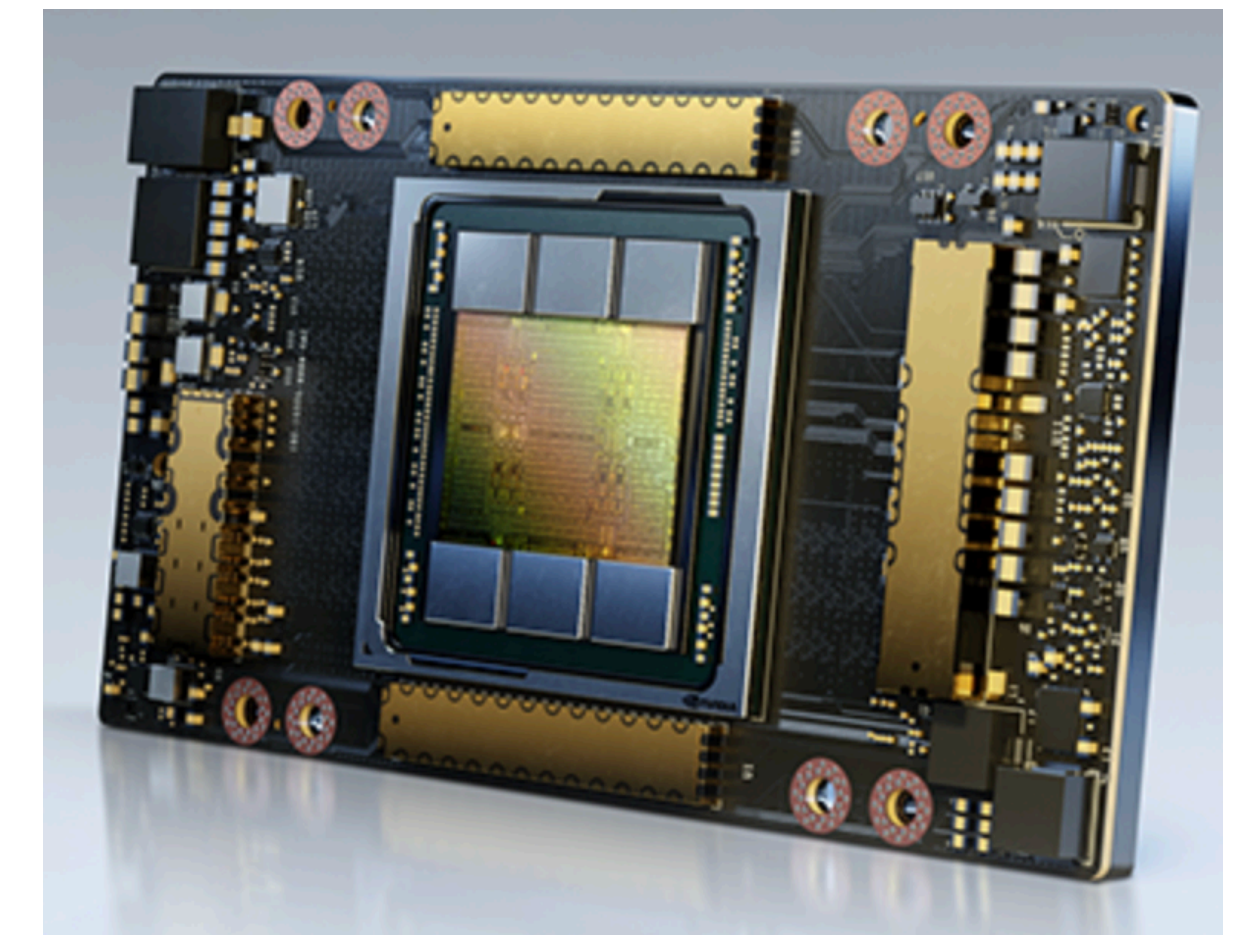
## GPU System Grete

- Most energy efficient system in Germany (top 12 in the world)
- 144 NVIDIA A100 GPUs with 40GB memory
- Multi-Instance GPU: Splitting one GPU in more GPUs possible; we will be using *splits*
- Between nodes InfiniBand (2x200 GBit/s per node)
- GPU-to-Storage 130TiB local flash-based storage

GPU Cluster Grete: [https://www.gwdg.de/documents/20182/27257/GN\\_3-2023\\_www.pdf](https://www.gwdg.de/documents/20182/27257/GN_3-2023_www.pdf)



**GPU System Grete**

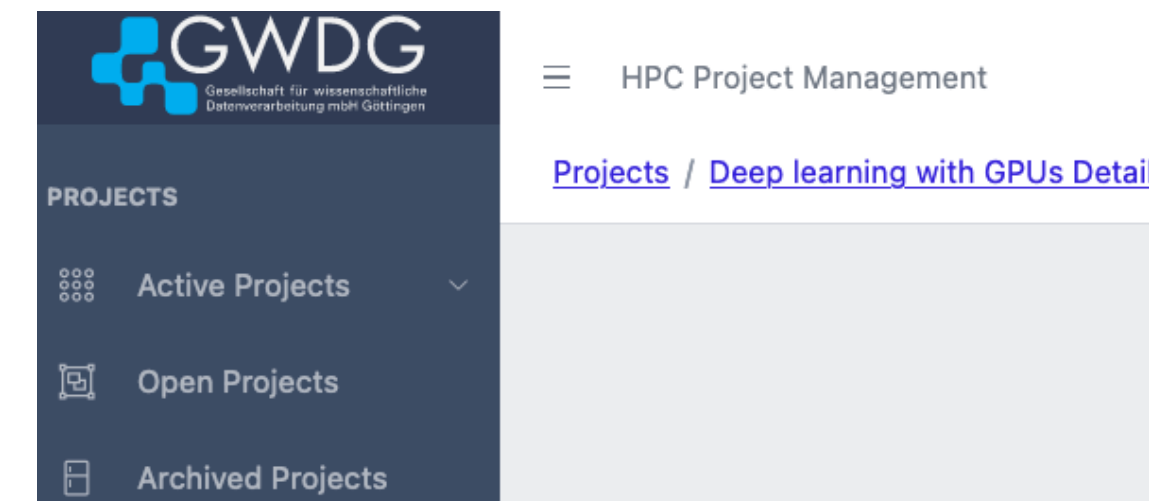
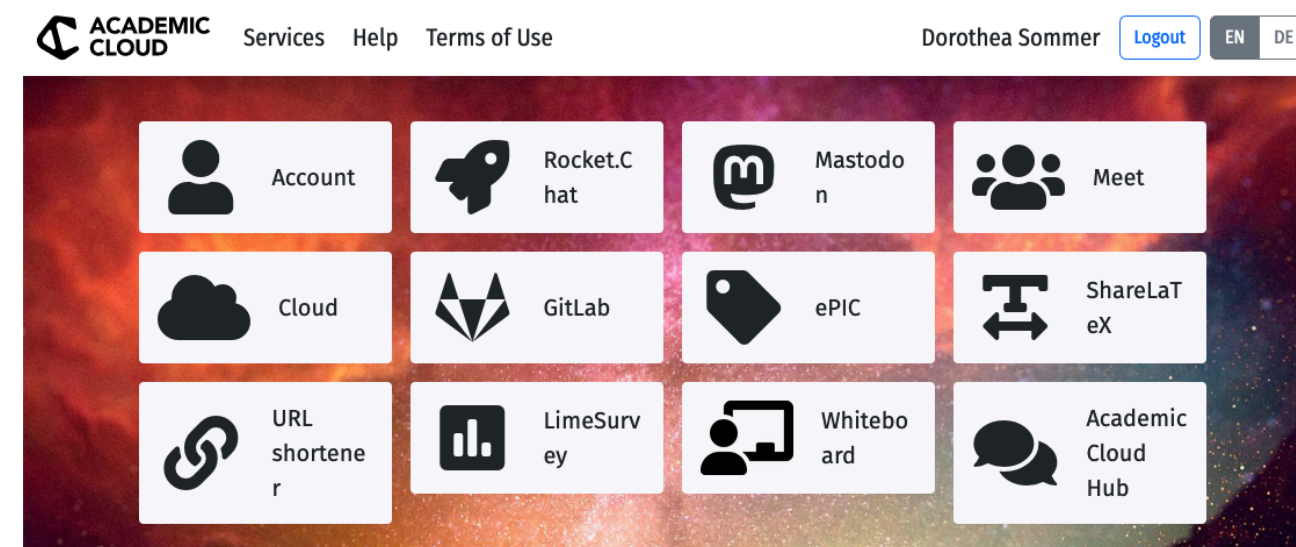


<https://www.nvidia.com/de-de/data-center/a100/>

**GPU A100**



# Cluster Access: This Session



- 1 Make an AcademicCloud account (students/researchers often have one!).

- 3 If new account: Send us your mail address!
- 4 You will get a **username** in an e-mail from the project portal from us.



- 2 Upload an **ssh-key** in:  
[id.academiccloud.de/security](https://id.academiccloud.de/security)

→ `ssh -i .ssh/yourkey username123@glogin9.hlrn.de`

# Cluster Access

## Free User Account

- 1 Under <https://zulassung.hlrn.de/> fill out the application for a user account.

NHR@ZIB NHR@GÖTTINGEN

Login | Logout

deutsch  | english 



### Joint Service Portal of NHR@ZIB and NHR@Göttingen (HLRN)



— Until the NHR-wide system (JARDS) is launched, please apply here for accounts and projects —

#### User Accounts

(General Information in a new window)

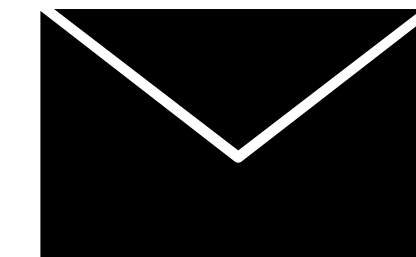
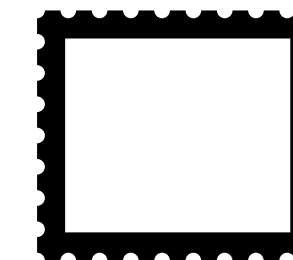
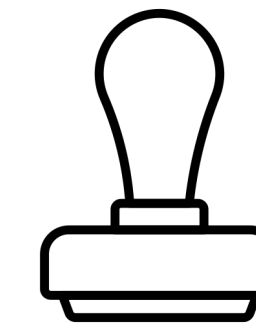
 **Application** for a user account  
 Help in window on the right hand side

 **Account information** (retrieve and modify data)  
(contact data, target hosts, query allocation and usage, password)  
 Help in window on the right hand side

 **Manage keys** for login to the HLRN  
(parallel usage of multiple keys is possible)  
 Help in window on the right hand side

<https://zulassung.hlrn.de/>

- 2 Print this form.  
Let your **university employer** (head of institute/supervisor) **sign this form** to confirm your position.



- 3 Send this signed form to the GWDG.
- 4 You will receive an e-mail with **your credentials**. This will take some days.

**Note:** If any instruction on [zulassung.hlrn.de](https://zulassung.hlrn.de) deviates from this procedure, please follow the current instructions on the website.

# Cluster Access

## Projects = More Compute

### Overview of Application Types

Application	Compute Time Grant	Compute time per quarter [coreh]	Previous Grants	Review process	Call	
Test Project/Easy Access	small projects or preparation	75k (up to 300k upon request)	none	automatically granted upon user account creation	rolling	<a href="#">Apply</a>
Compute Project	Normal	300k - 5M	none	Scientific board of NHR@ZIB, NHR@Göttingen	quarterly	cf. below
			DFG/BMBF /NHR/GCS/EU	Whitelist: simplified review by Scientific board		
	Large-scale ("Großprojekt")*	>=5M	none	Scientific board + NHR panel ("Nutzungsausschuss")		
			DFG/BMBF /NHR/GCS/EU	Whitelist: simplified review + NHR panel		

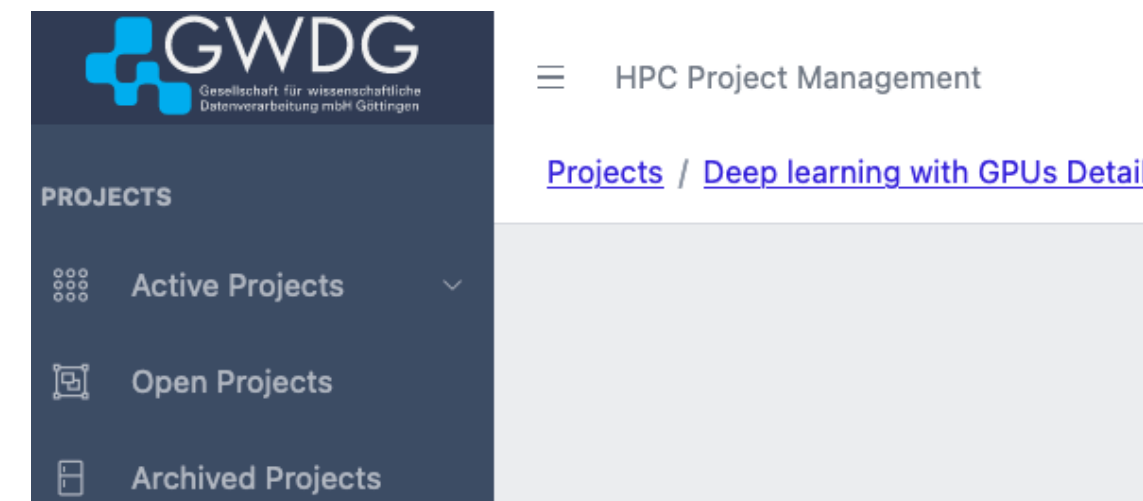
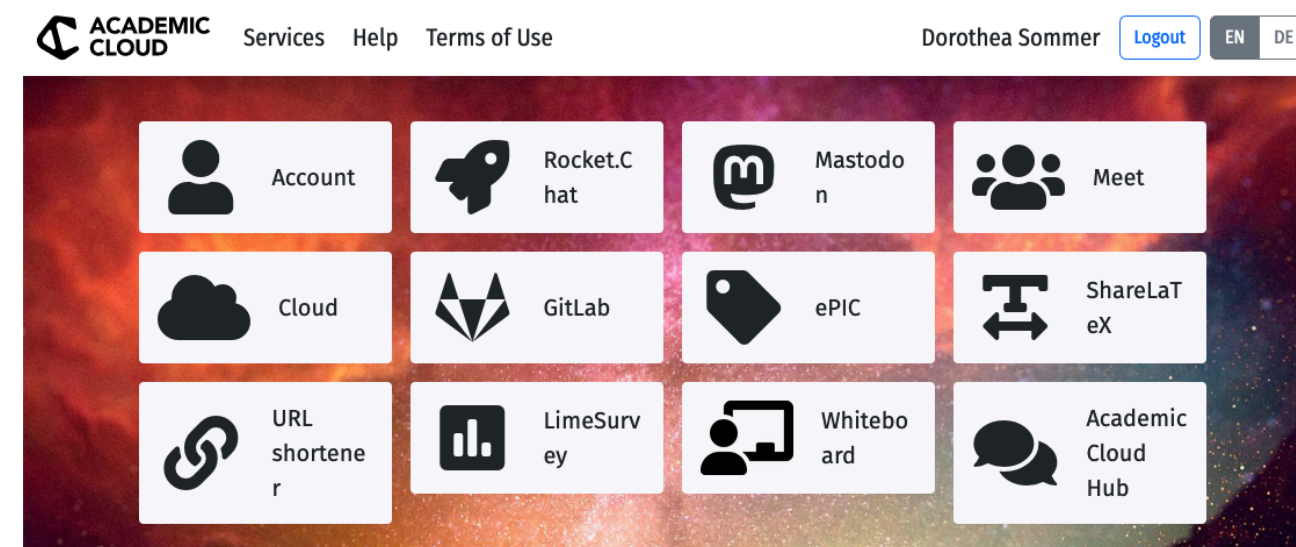
- 0 Free compute when you sign up.
- 1 Apply for a compute project at <https://zulassung.hlrn.de/>. It will be judged by scientific impact and compute hours needed. Deadlines every 3 month. **Free** for researchers in Germany.

\* Please note that the application process for Normal and Large-scale projects is the same. Large-scale projects will be identified by the Scientific board.

<https://www.hlrn.de/doc/display/PUB/Apply+for+a+Compute+Project>



# Cluster Access: This Session



- 1 Make an AcademicCloud account (students/researchers often have one!).

- 3 If new account: Send us your mail address!
- 4 You will get a **username** in an e-mail from the project portal from us.



- 2 Upload an **ssh-key** in:  
[id.academiccloud.de/security](https://id.academiccloud.de/security)

→ `ssh -i .ssh/yourkey username123@glogin9.hlrn.de`

# Training Checklist

## Tools & Infrastructure

- **Place to train (access to cluster)**
  - Get free access for Emmy as a researcher in Germany.

- **Cluster essentials**

- **Data**

- **Code**

- **Monitoring & tracking**

Login

Compute Nodes

User Side

frontend

ssh



user

1 Log in on the  
frontend (glogin9).

compute  
nodes



Few frontend nodes (glogin9)  
... and a lot of compute nodes.

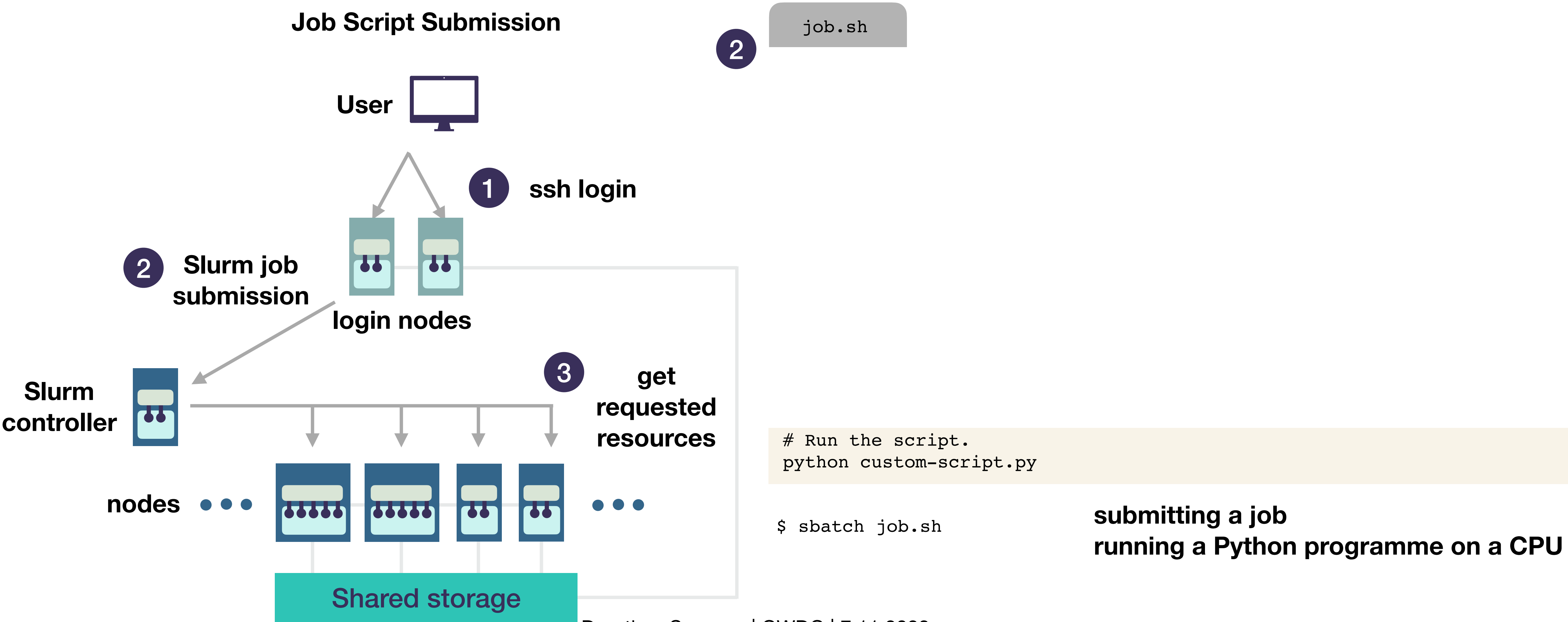
GPUS are here!

2 Do *not* directly run your jobs on the frontend!  
Use a scheduler, which will pass your programmes to  
the compute nodes.

# Cluster Essentials

## Slurm in 1 minute

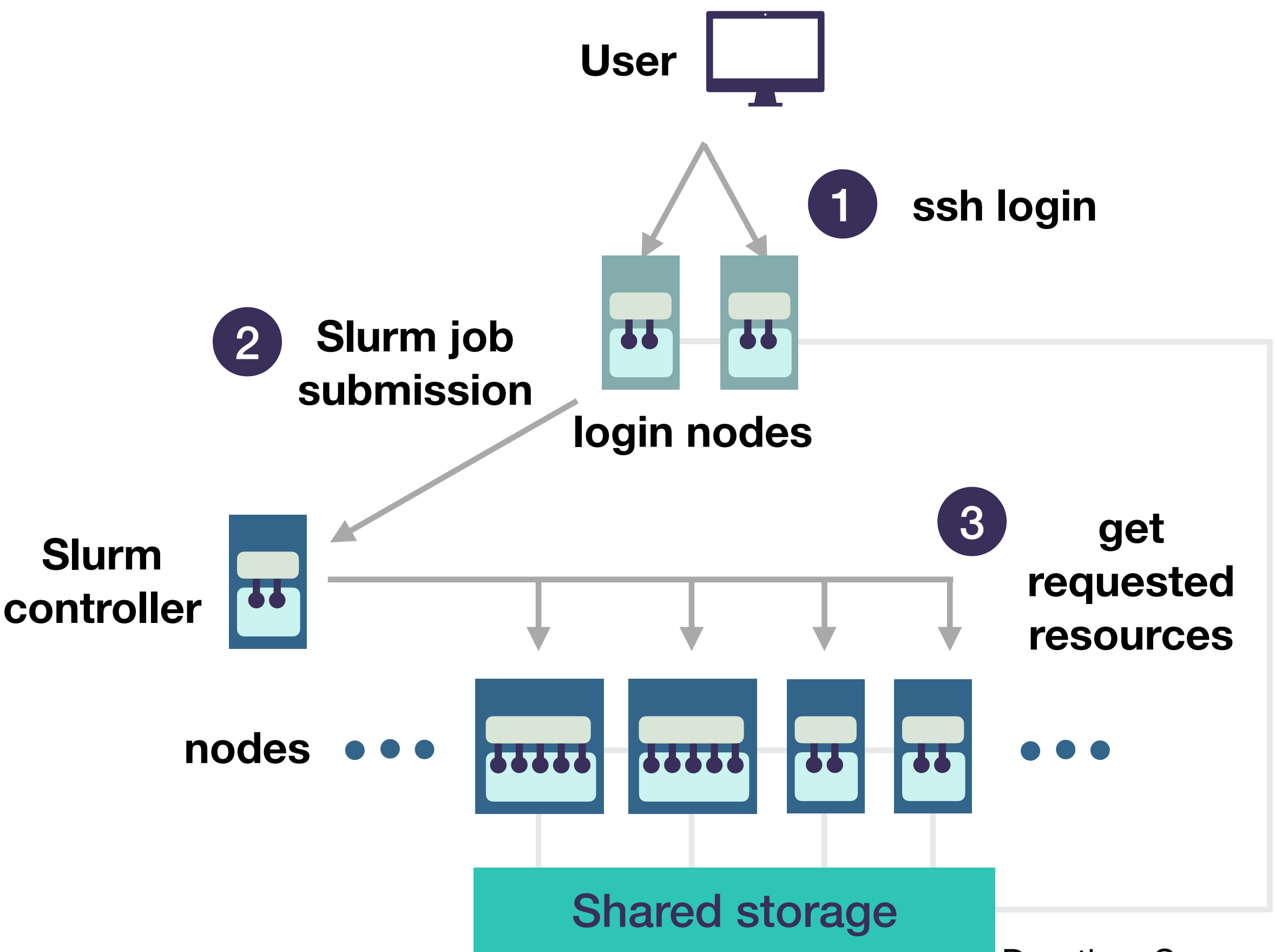
### Job Script Submission



# Cluster Essentials

## Slurm in 1 minute

### Job Script Submission



2

```
job.sh

#!/bin/bash
#SBATCH --job-name=custom-name

#SBATCH --nodes=2                # total number of nodes
#SBATCH --ntasks-per-node=1      # total number of tasks per node
#SBATCH --cpus-per-task=2        # cpu cores per task, > 1
                                # but less than cores per node
#SBATCH --mem-per-cpu=4GB        # memory given to each cpu

#SBATCH --time=00:01:00          # total run time limit (HH:MM:SS)
#SBATCH --mail-type=all          # mail when job begins and ends
#SBATCH --mail-user=custom@mail.com

# Run the script.
python custom-script.py
```

resources you need

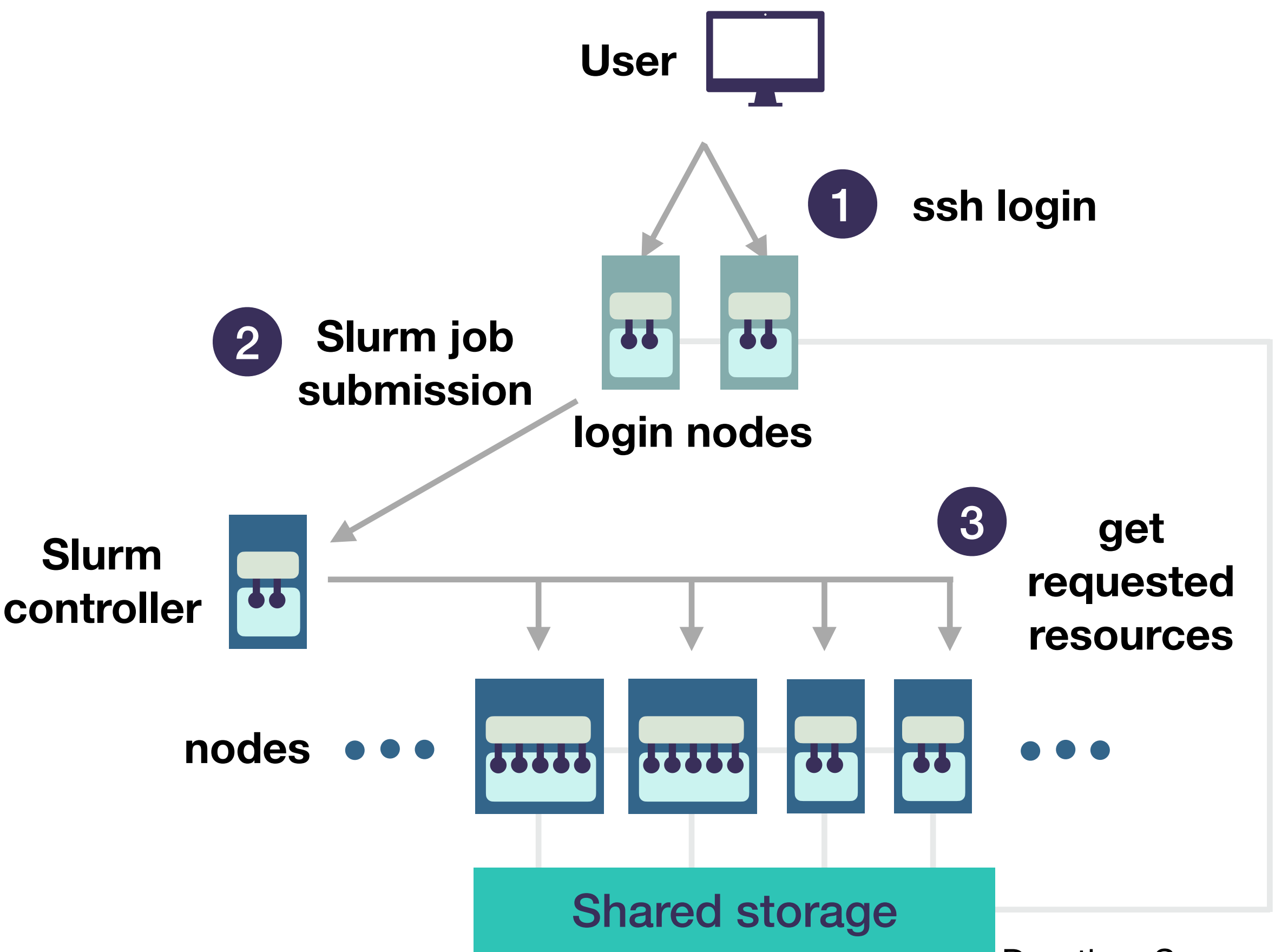
\$ sbatch job.sh

submitting a job  
running a Python programme on a CPU

# Cluster Essentials

## Slurm in 1 minute

### Job Script Submission



### resources you need

2

```
job.sh

#!/bin/bash
#SBATCH --job-name=custom-name

#SBATCH --nodes=2                # total number of nodes
#SBATCH --ntasks-per-node=1      # total number of tasks per node
#SBATCH --cpus-per-task=2        # cpu cores per task, > 1
                                # but less than cores per node
#SBATCH --mem-per-cpu=4GB        # memory given to each cpu

#SBATCH --time=00:01:00          # total run time limit (HH:MM:SS)
#SBATCH --mail-type=all          # mail when job begins and ends
#SBATCH --mail-user=custom@mail.com

# Prepare the environment.
module load anaconda3/2021.05
source activate custom-env

# Run the script.
python custom-script.py
```

\$ sbatch job.sh

submitting a job  
running a Python programme on a CPU



# Training Checklist

## Tools & Infrastructure

- **Place to train (access to cluster)**
  - Get free access for Emmy as a researcher in Germany.
- **Cluster essentials**
  - All jobs need to go through a scheduler (here: Slurm scheduler).
- **Data**
- **Code**
- **Monitoring & tracking**

Login

Compute Nodes

User Side

frontend

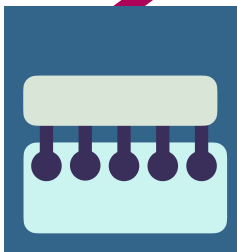
ssh



user

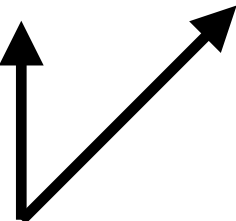


compute  
nodes



1

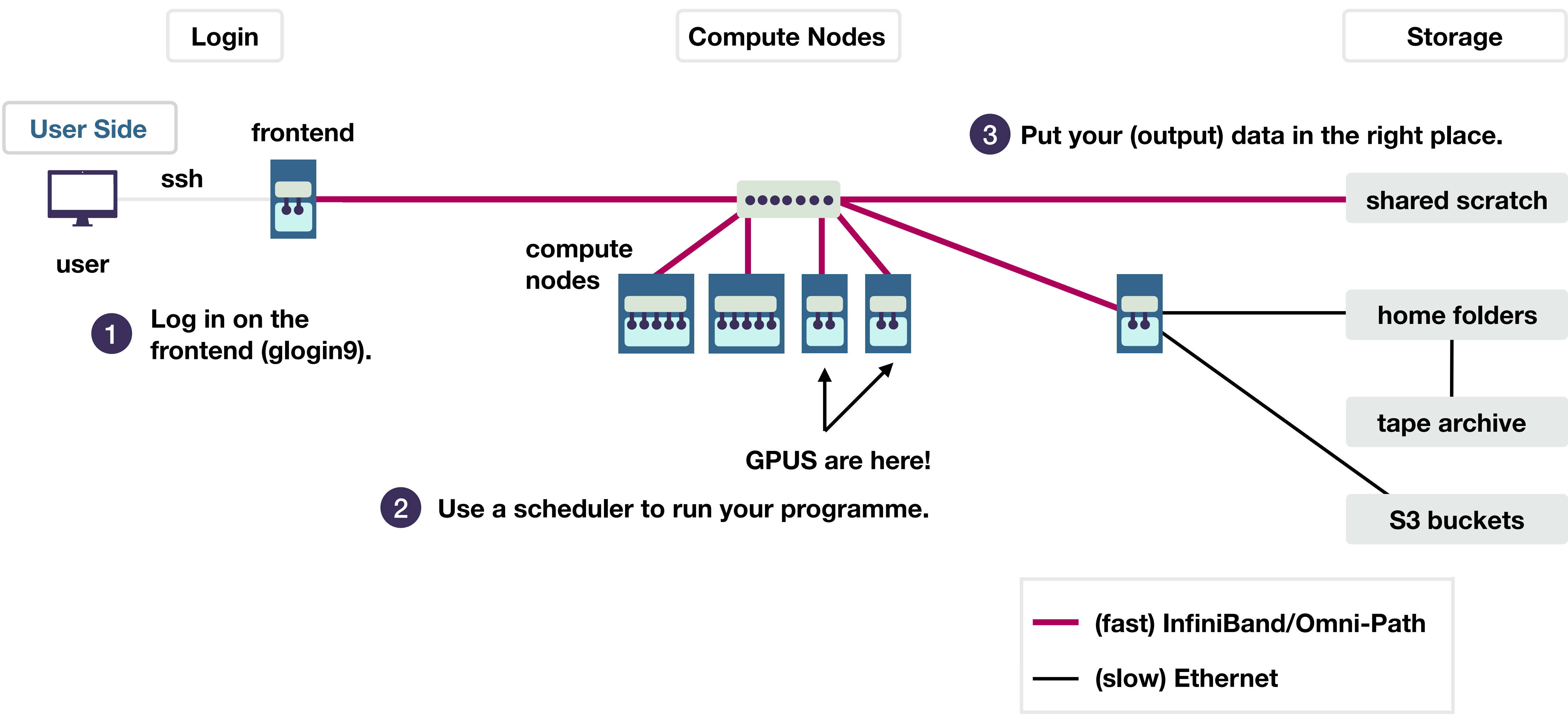
Log in on the  
frontend (glogin9).



GPUS are here!

2

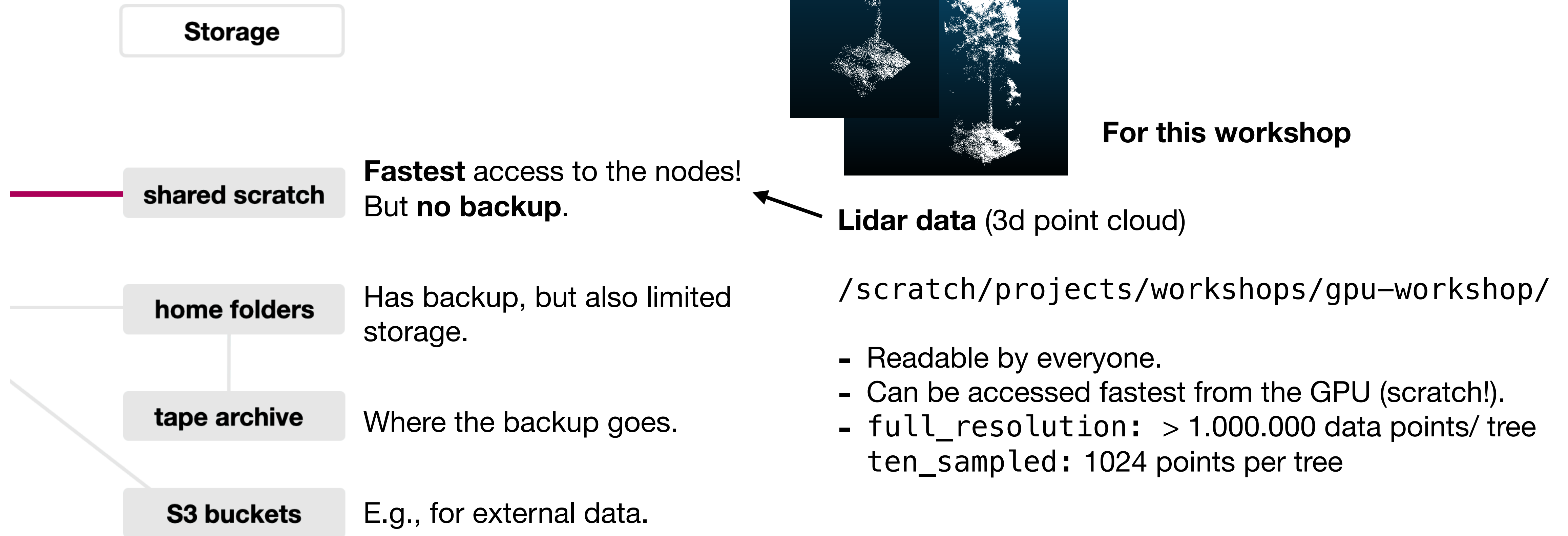
Use a scheduler to run your programme.





# Data

## Where to store your data?



# Training Checklist

## Tools & Infrastructure

- **Place to train (access to cluster)**
  - Get free access for Emmy as a researcher in Germany.
- **Cluster essentials**
  - All jobs need to go through a scheduler (here: Slurm scheduler).
- **Data**
  - Use /scratch/ for the fastest access, but note that it has no backup.
- **Code**
- **Monitoring & tracking**

# Environments



## Why is environments is important?

- **Reproducibility!**
- Projects require different Python and package versions.
- On the cluster, everyone needs a different environment.

- 1 Create a new conda environment.
- 2 Install all packages, either manually or from the `requirements.txt`
- 3 Activate environment before running a job.

```
requirements.txt 2.01 KIB
1 anyio==3.6.2
2 argon2-cffi==21.3.0
3 argon2-cffi-bindings==21.2.0
4 asttokens==2.2.0
5 attrs==22.1.0
6 backcall==0.2.0
7 beautifulsoup4==4.11.1
8 bleach==5.0.1
9 certifi==2022.9.24
10 cffi==1.15.1
11 charset-normalizer==2.1.1
12 comm==0.1.1
13 contourpy==1.0.6
14 cycler==0.11.0
15 debugpy==1.6.4
16 decorator==5.1.1
```



# Cluster Essentials

## Slurm in 1 minute

2

job.sh

: resources you need

```
#!/bin/bash
#SBATCH --job-name=custom-name

#SBATCH --nodes=2                # total number of nodes
#SBATCH --ntasks-per-node=1      # total number of tasks per node
#SBATCH --cpus-per-task=2        # cpu cores per task, > 1
                                # but less than cores per node
#SBATCH --mem-per-cpu=4GB        # memory given to each cpu

#SBATCH --time=00:01:00          # total run time limit (HH:MM:SS)
#SBATCH --mail-type=all          # mail when job begins and ends
#SBATCH --mail-user=custom@mail.com
```

```
# Prepare the environment.
module load anaconda3/2021.05
source activate custom-env
```

```
# Run the script.
python custom-script.py
```

\$ sbatch job.sh

**submitting a job**  
**running a Python programme on a CPU**

# Frameworks

The frameworks are similar: define a model in Python code, optimised computations in the background



Josh Tobin  
@josh\_tobin\_

Why do people always ask what ML framework to use?  
It's easy:

- jax is for researchers
- pytorch is for engineers
- tensorflow is for boomers

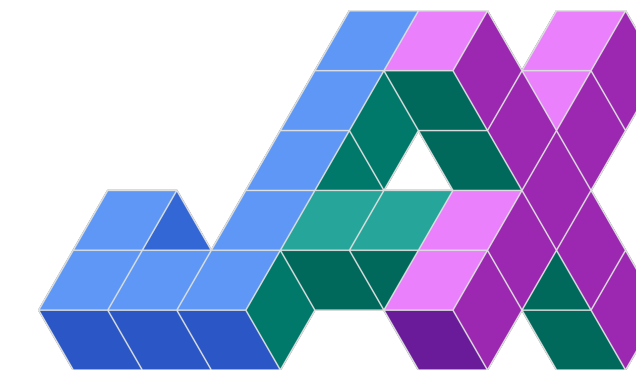
6:24 PM · Mar 11, 2021 · Twitter Web App



- From Google (2015)



- From Facebook, now part of Linux foundation
- Dominant: competitions (77% winners), models, number of papers



- Newest: from Google v.0.3.13 (2022)
- Auto-differentiation, vectorisation
- Deep learning: needs separate framework (Flax, Haiku)

# Code



- 1 Log in to the frontend **glogin9**.
- 2 Clone the code you need (e.g. in your home directory).

```
utils.py  train.py 6 x  model.py  $ submit_train.sh
train.py > create_data_loader
112     return model
113
114 if __name__ == "__main__":
115     print("Start training")
116
117     ### LEARNING PARAMETERS ###
118
119     n_classes = len(TREE_SPECIES)
120     device = torch.device("cuda" if torch.cuda.is_available() else "cpu") # GPU available?
121     print(f"Training with: {device}")
122
123     saved_models_path = "./saved_models"
124     if not os.path.exists(saved_models_path):
125         os.makedirs(saved_models_path)
126         print(f"Created path for models in {saved_models_path}")
127
128     learning_rate = 0.001
129     batch_size = 32
130     num_training_epochs = 100
131
132
```

<https://gitlab-ce.gwdg.de/dmuelle3/deep-learning-with-gpu-cores>



# Training Checklist

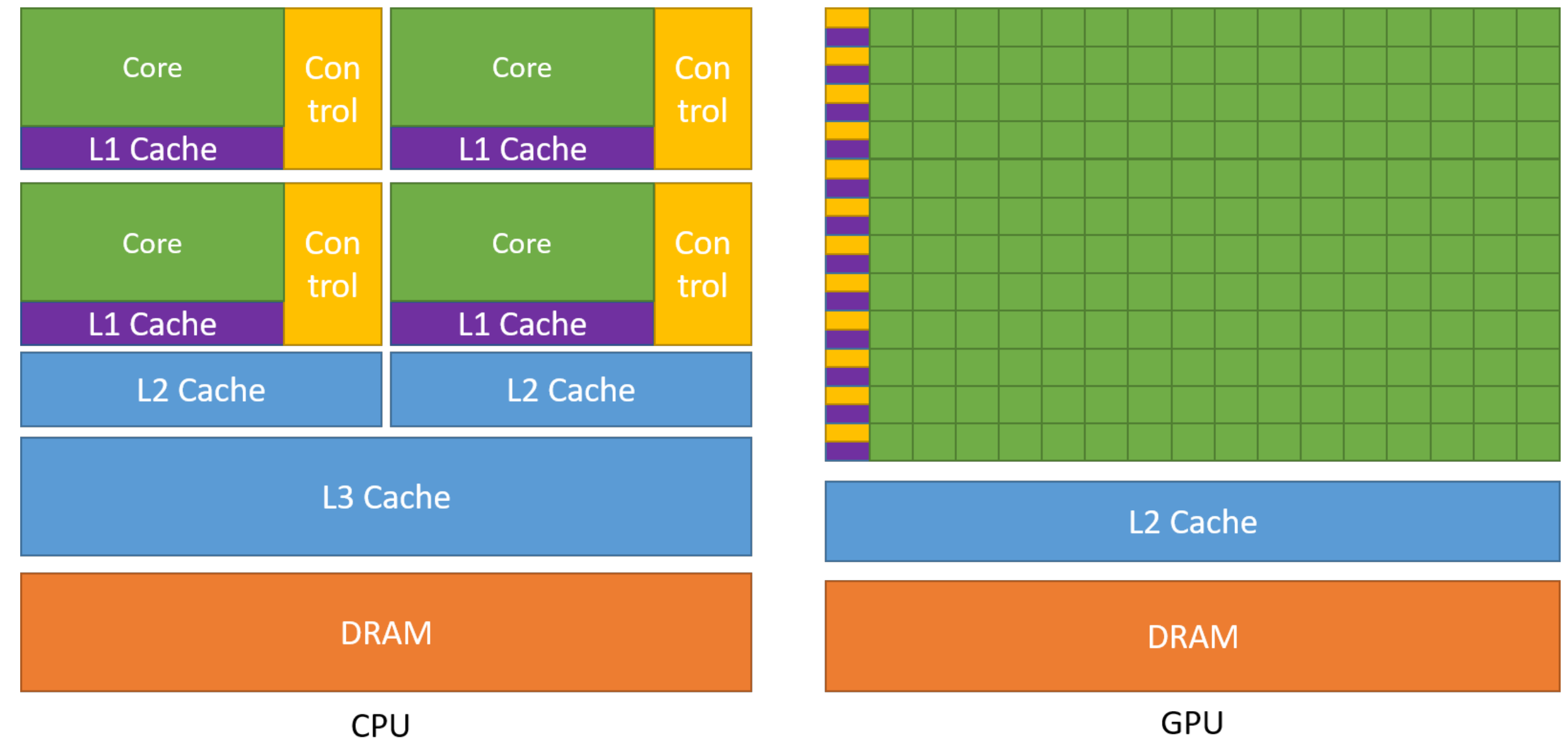
## Tools & Infrastructure

- **Place to train (access to cluster)**
  - Get free access for Emmy as a researcher in Germany.
- **Cluster essentials**
  - All jobs need to go through a scheduler (here: Slurm scheduler).
- **Data**
  - Use `/scratch/` for the fastest access, but note that it has no backup.
- **Code**
  - Choose framework (PyTorch).
  - Make or clone GitHub repository.
  - Create a conda environment with `requirements.txt`.
- **Monitoring & tracking**

# Monitoring

## Basic GPU ideas: For what are GPUs efficient?

- Sequential operations are called a thread.
- GPUs are efficient at running the same operation on a large number of elements (i.e., running a lot of threads simultaneously).



Internal comparison between a CPU and a GPU.

<https://docs.nvidia.com/cuda/cuda-c-programming-guide/>

# Monitoring

## Basic GPU ideas: What to monitor?

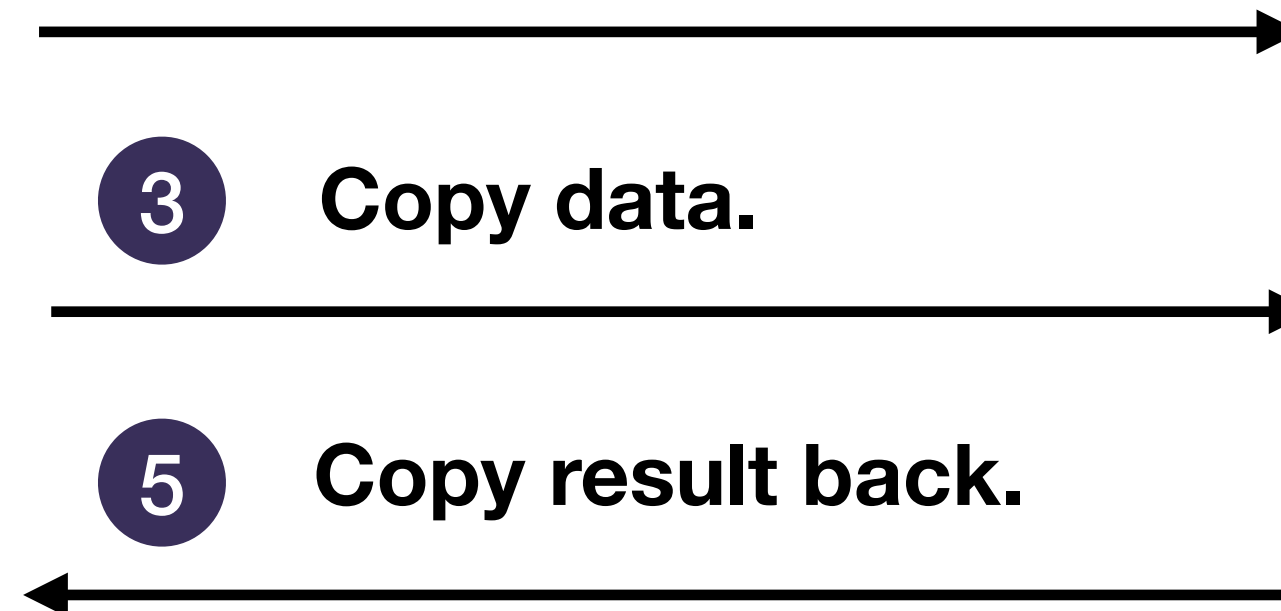
- 1 Start the program.  
Define the network.



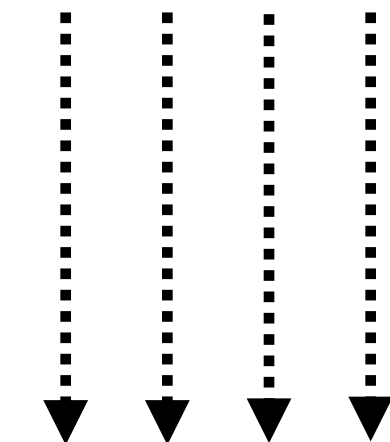
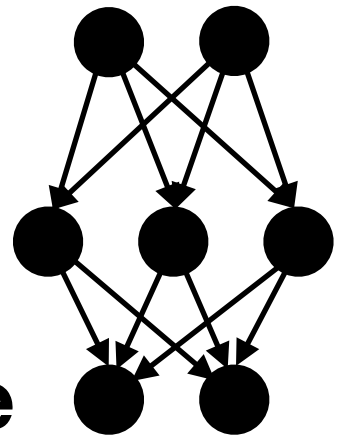
- 2 Copy model.

- 3 Copy data.

- 5 Copy result back.



- 4 Compute in threads.



“**Host**” = primary processor  
that manages the copying  
and controls the GPU

Computation is done  
in a **kernel function** that is  
executed in **parallel**  
**simultaneously** among  
many threads.

Same operation, just  
different data for the nodes.

# Monitoring

## Basic GPU ideas: What to monitor?

- 1 Start the program.  
Define the network.



“**Host**” = primary processor  
that manages the copying  
and controls the GPU

- 2 Copy model.



- 3 Copy data.



- 5 Copy result back.



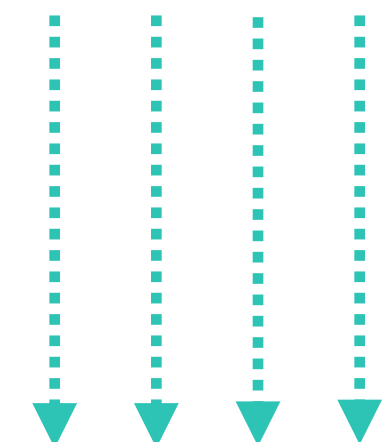
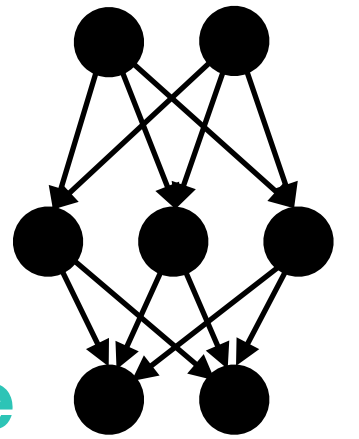
=> **Bandwidth matters!**

=> **Make sure to compute a lot,  
not only copy a lot!**

=> **Size (GB) matters!**



- 4 Compute  
in threads.



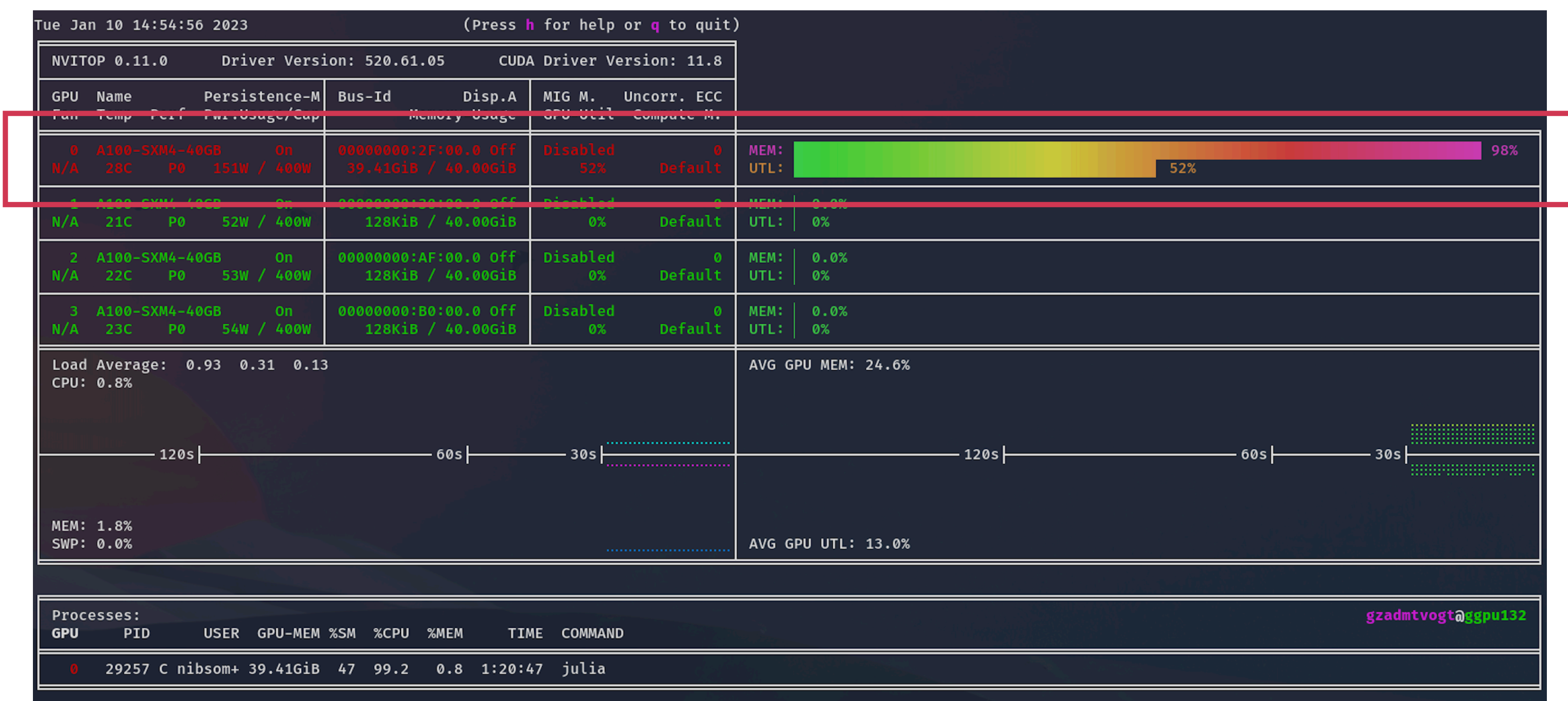
Computation is done  
in a **kernel function** that is  
executed in **parallel**  
**simultaneously** among  
many threads.



# Monitoring

## Basic concept: How to monitor?

- 1 ssh to the node your jobs runs on.
- 2 Look up the memory (MEM) and utilisation (UTL) with module `load nvitop nvitop`.



Example output from running `nvitop`.

# Training Checklist

## Tools & Infrastructure

- **Place to train (access to cluster)**
  - Get free access for Emmy as a researcher in Germany.
- **Cluster essentials**
  - All jobs need to go through a scheduler (here: Slurm scheduler).
- **Data**
  - Use `/scratch/` for the fastest access, but note that it has no backup.
- **Code**
  - Choose framework (PyTorch).
  - Make or clone GitHub repository.
  - Create a conda environment with `requirements.txt`.
- **Monitoring & tracking**
  - Ensure that your jobs utilise the GPU well (computation and memory).

# What we'll do

	Deep Learning with GPU cores	
09.30 - 09.45	Welcome	
09.45 - 10.15 <b>(30 min)</b>	Deep Learning and Infrastructure	Learn how to train a neural network with a GPU.
10.15 - 11.30 <b>(60 min)</b>	Practical: Working on the GPU	
11.30 - 11.45	Short break ☕	
11.45 - 12.00 <b>(15 min)</b>	Introduction to Profiling	Learn how to profile the training and training efficiently.
12.00 - 12.45 <b>(45 min)</b>	Practical: Profiling Jobs	
12.45 - 13.00	General Q&A	

# **Practical Part I**

**Let's switch to the code...**



**Additional Material**

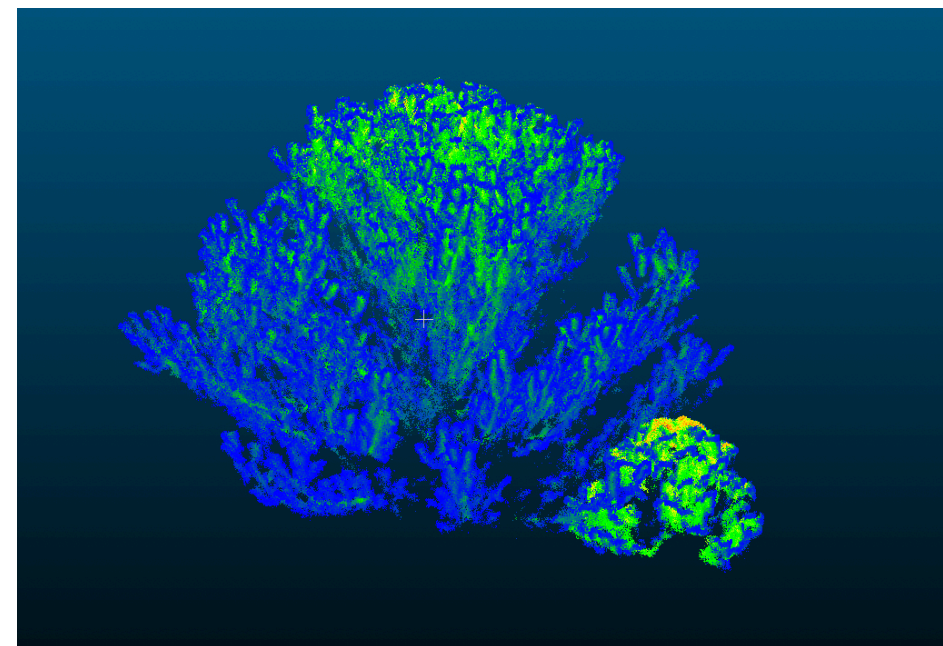
# Machine Learning Paradigms

	<div>Supervised<div>learning with teacher</div></div>	<div>Unsupervised<div>learning representations</div></div>	<div>Reinforcement<div>learning behaviour</div></div>
Data	Observations $\underline{x}_1, \dots, \underline{x}_n$ Labels $y_1, \dots, y_n$	Observations $\underline{x}_1, \dots, \underline{x}_n$	States $\underline{s}_1, \dots, \underline{s}_n$ Actions $a_1, \dots, a_n$ Rewards $r_1, \dots, r_n$
Aim	<div><math>\underline{x}_i \longrightarrow</math><div>Model</div><math>\longrightarrow y_i</math></div> <div>predict label of observation</div> <div>regression, classification</div>	<div><math>\underline{x}_i \longrightarrow</math><div>Model</div><math>\longrightarrow x'_i</math></div> <div>extract relevant structures for useful representation</div> <div>dimensionality reduction, clustering</div>	<div><math>\underline{s}_i \longrightarrow</math><div>Model</div><math>\longrightarrow a_i</math></div> <div>find best action in every state</div>

# Unboxing the Model

## What do we know about our input data?

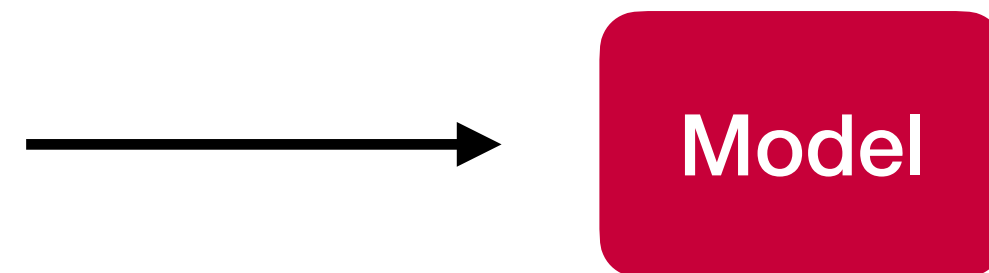
In **general**: the type of neural network depends on the **input data type**



3D point clouds

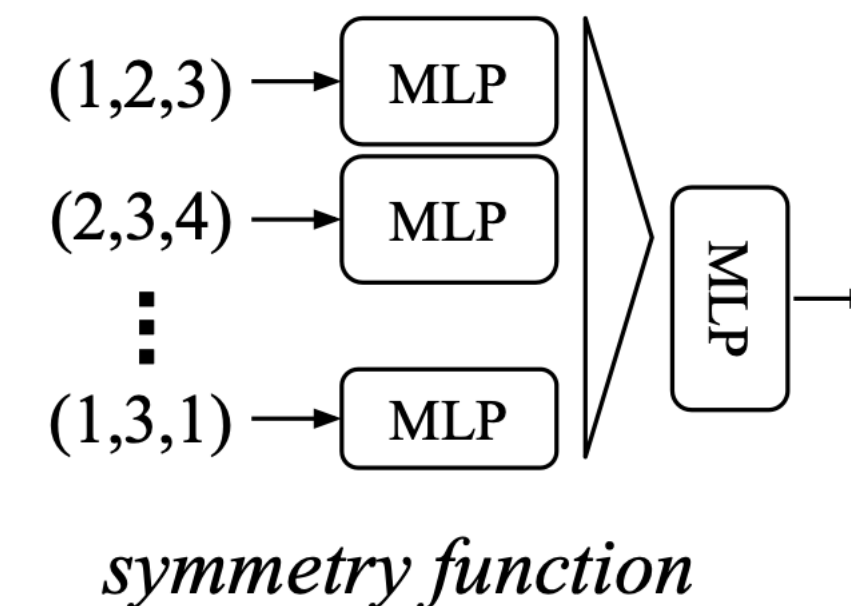
1. **unordered** set of points, a list of (x, y, z)
2. **invariance** under rigid transformations

$$f(x_1, \dots, x_n) \approx g(h(x_1), \dots, h(x_n))$$



$$h: \mathbb{R}^N \rightarrow \mathbb{R}^K \text{ neural network}$$

$$g: \underbrace{\mathbb{R}^K \times \dots \times \mathbb{R}^K}_n \rightarrow \mathbb{R} \text{ symmetric function (e.g., max pooling)}$$

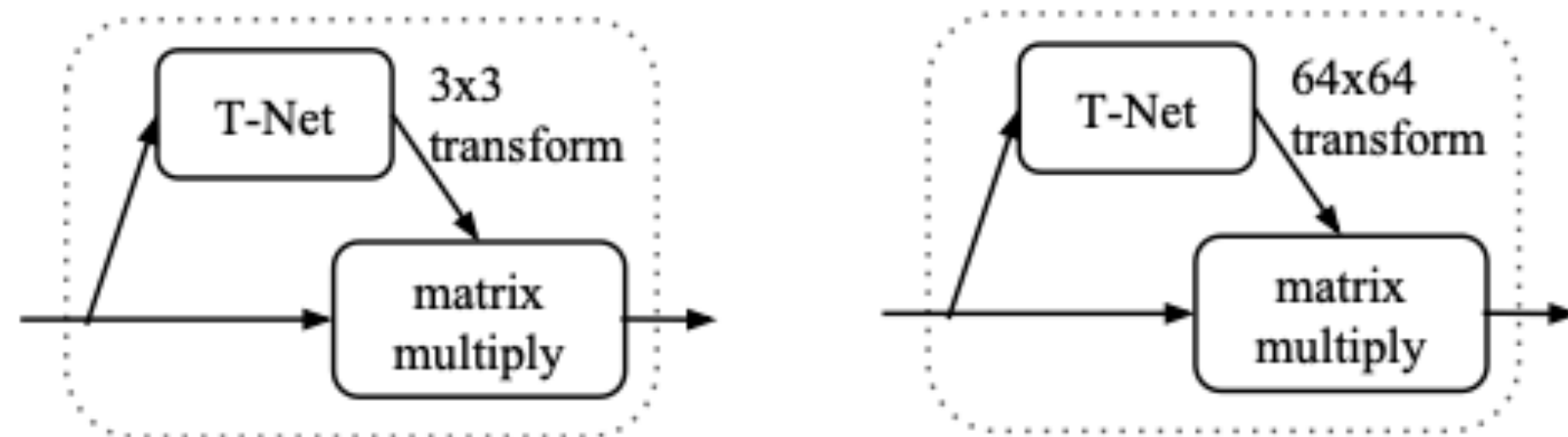
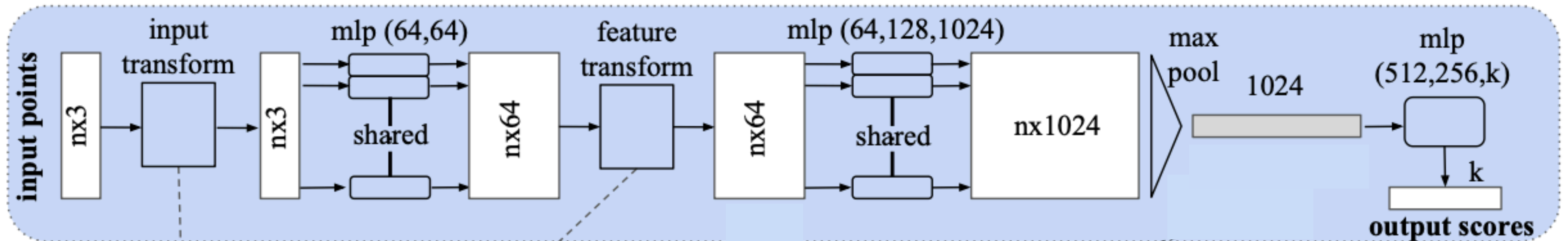


Qi et al. (2017) **PointNet**: Deep Learning on Point Sets for 3D Classification and Segmentation

# Unboxing the Model

## PointNet Architecture

*Classification Network*



**T-Net** as a learned affine transformation matrix

Look into the paper for more details.

Qi et al. (2017) **PointNet**: Deep Learning on Point Sets for 3D Classification and Segmentation



# Stack for Deep Learning

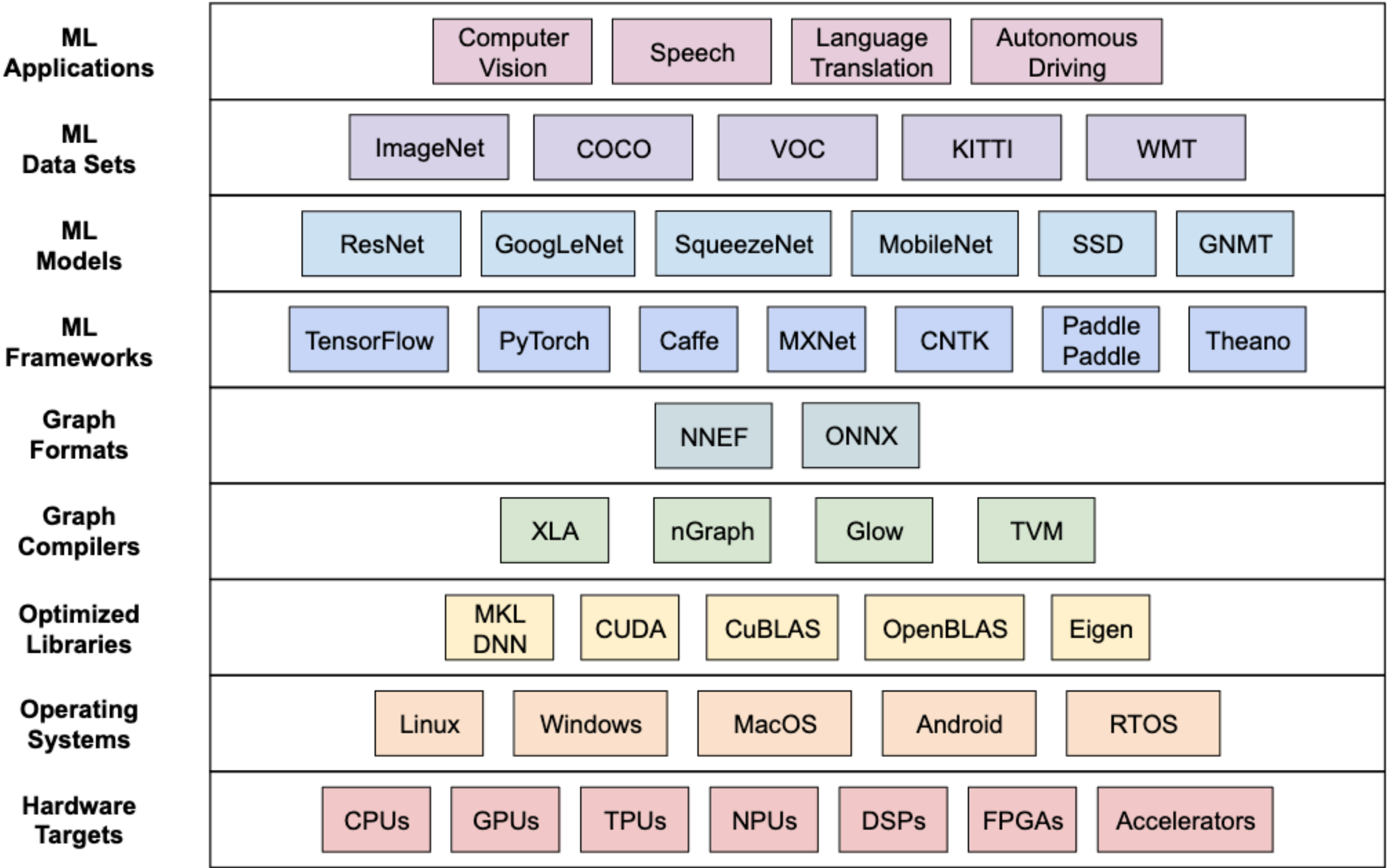
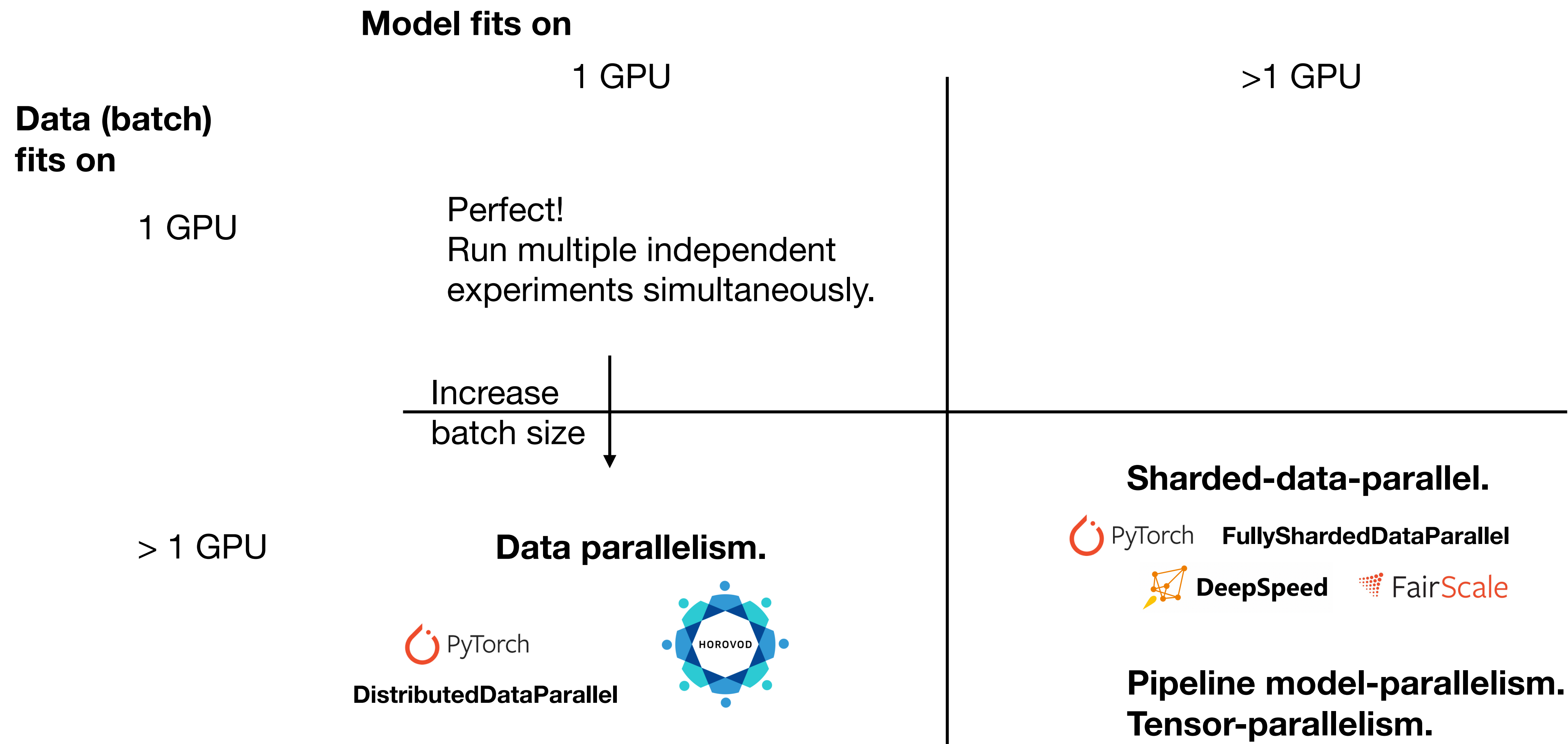
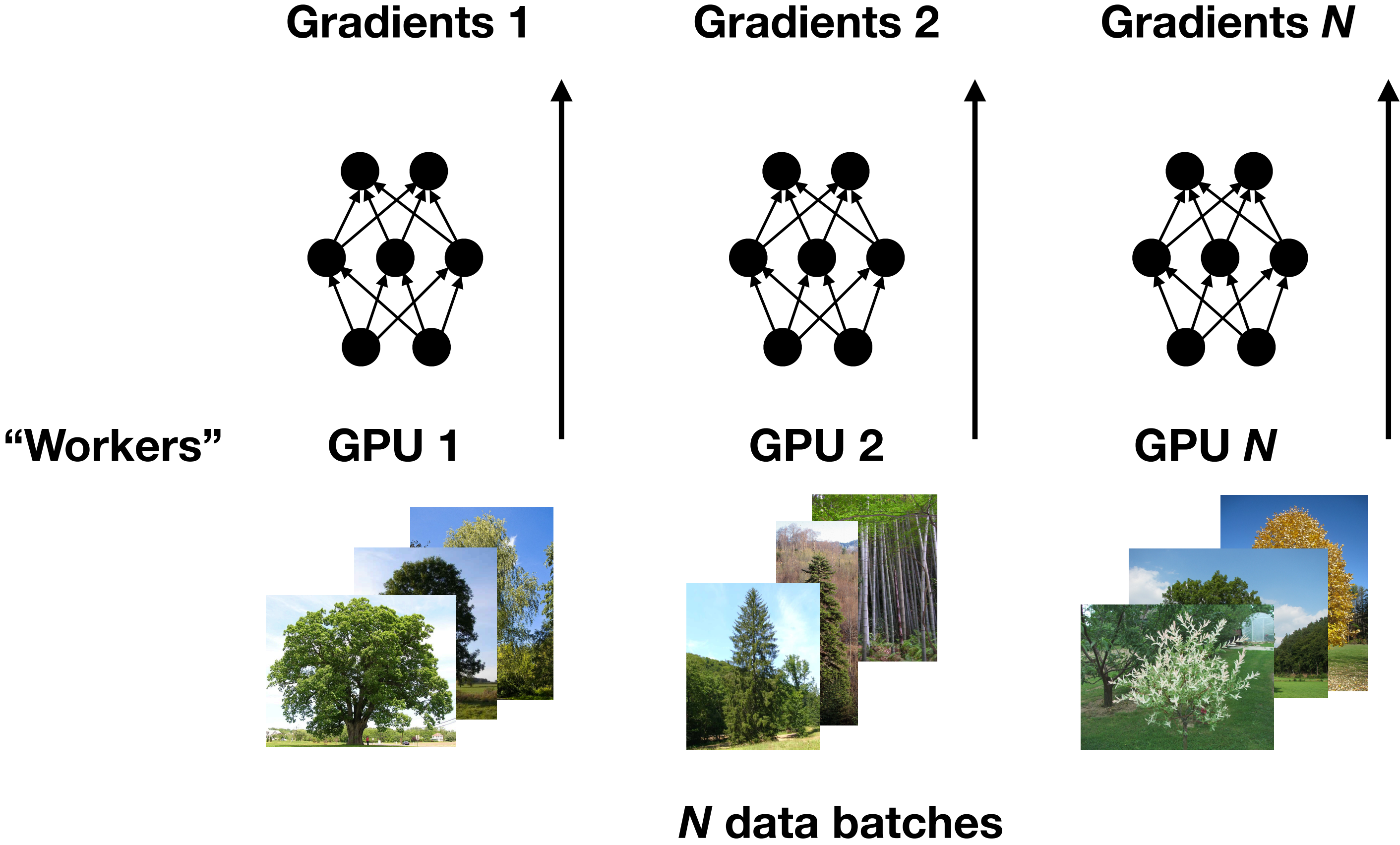


Fig. 2. The diversity of options at every level of the stack, along with the combinations across the layers, make benchmarking inference systems hard.

# Memory issues: What to do, if ...

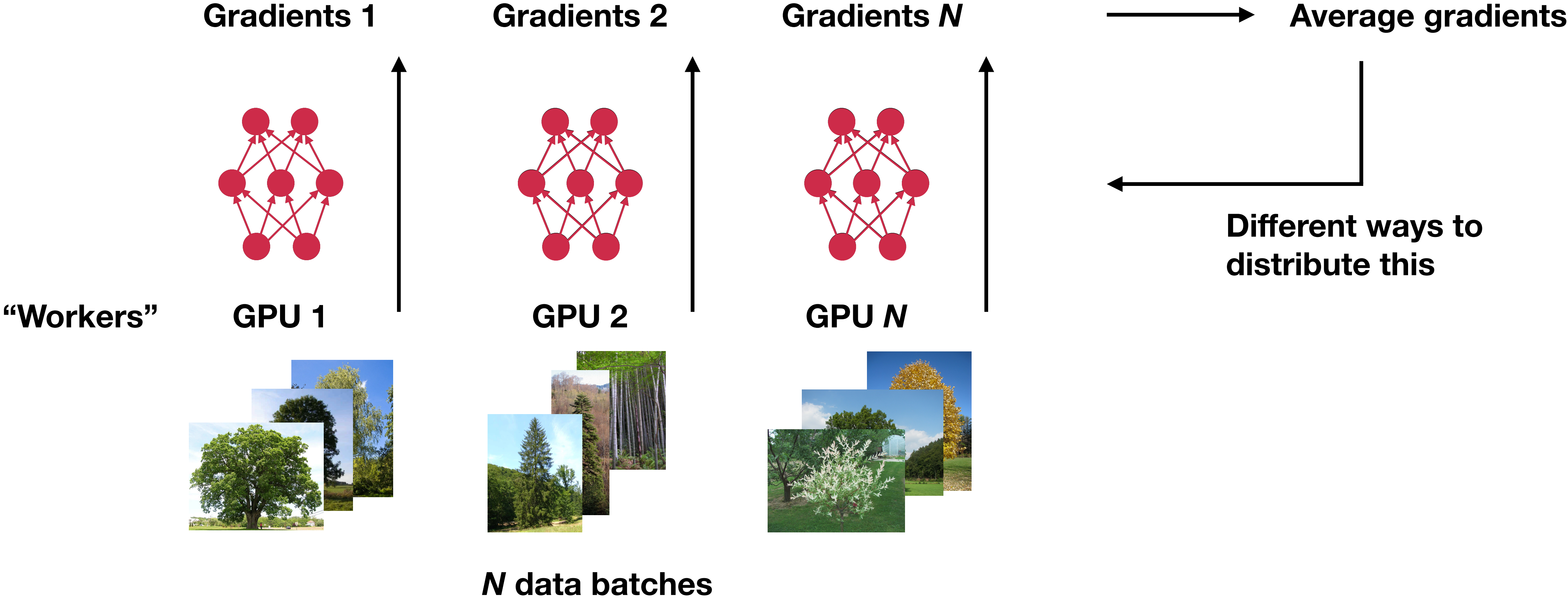


# Data Parallelism





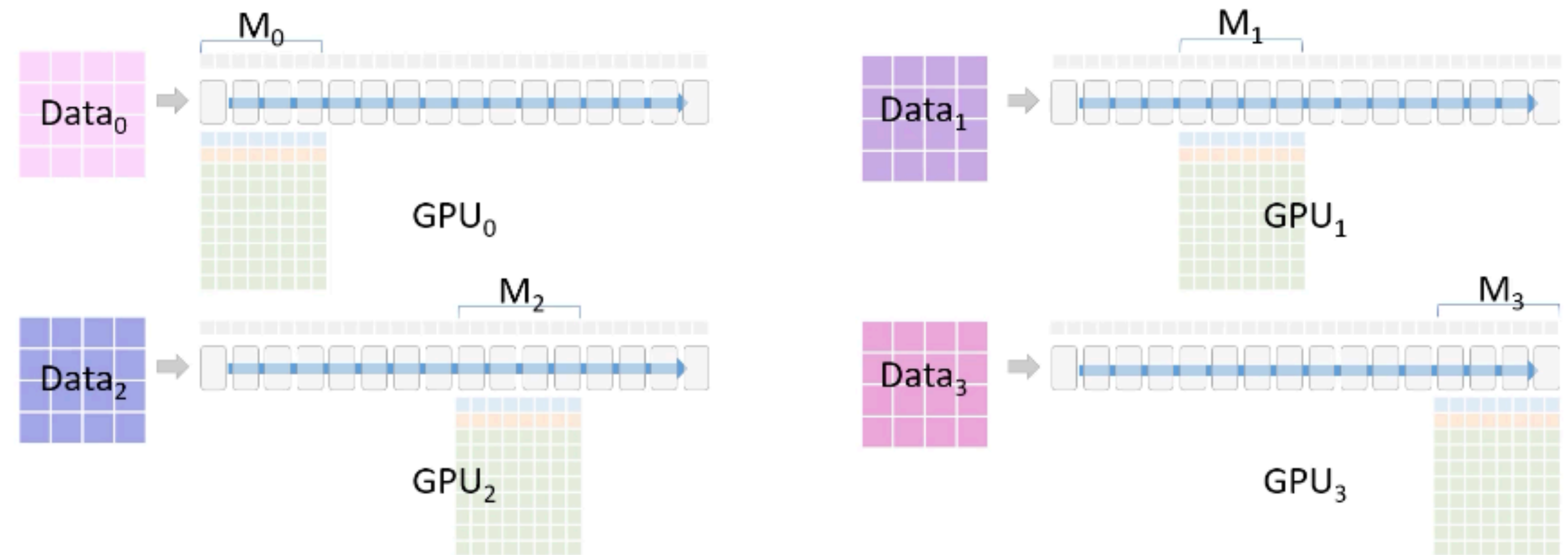
# Data Parallelism





# Sharded data-parallel

- Idea: optimizer states take most of model GPU memory
- copy model parameters around, only 1 GPU keeps optimiser states for 1 part of the model
- data is also sharded



Each GPU is responsible for 1 piece of the end model  
ZeRO P<sub>os+g+p</sub> and Gradient accumulation are used with the 4-way data parallelism

<https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/>

# Pipeline model-parallelism

- Idea: (processed) data is moved around between the GPUs
- Needs fine-tuning, otherwise only 1 GPU active at a time (and others are idle)

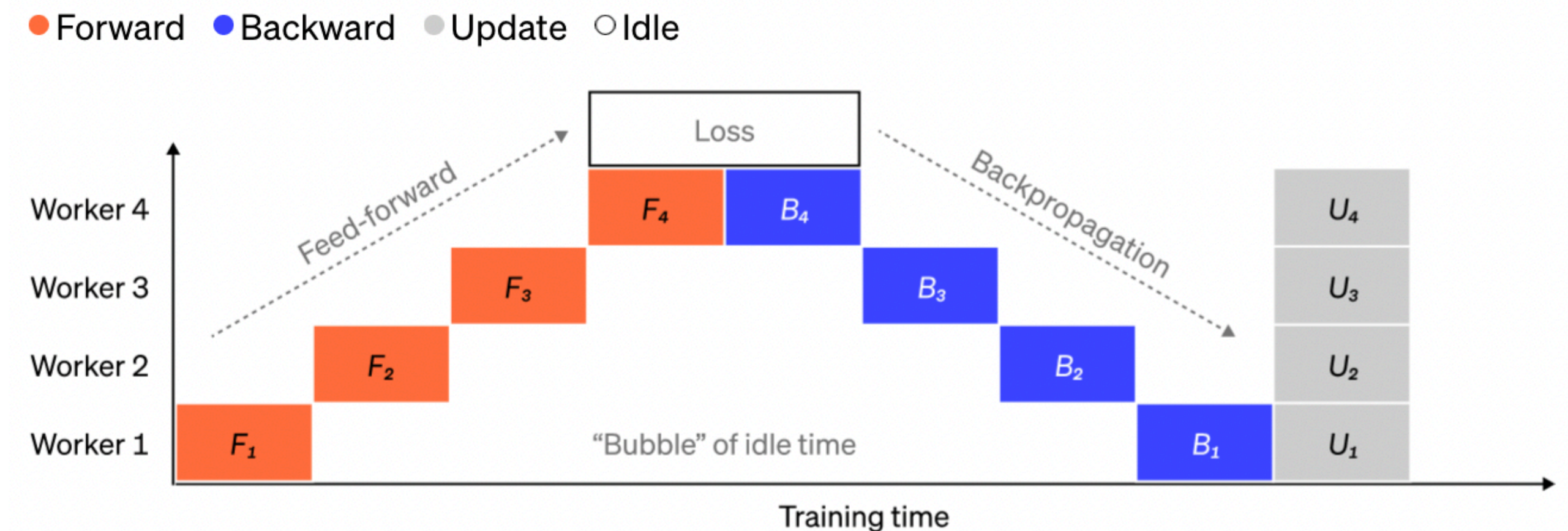
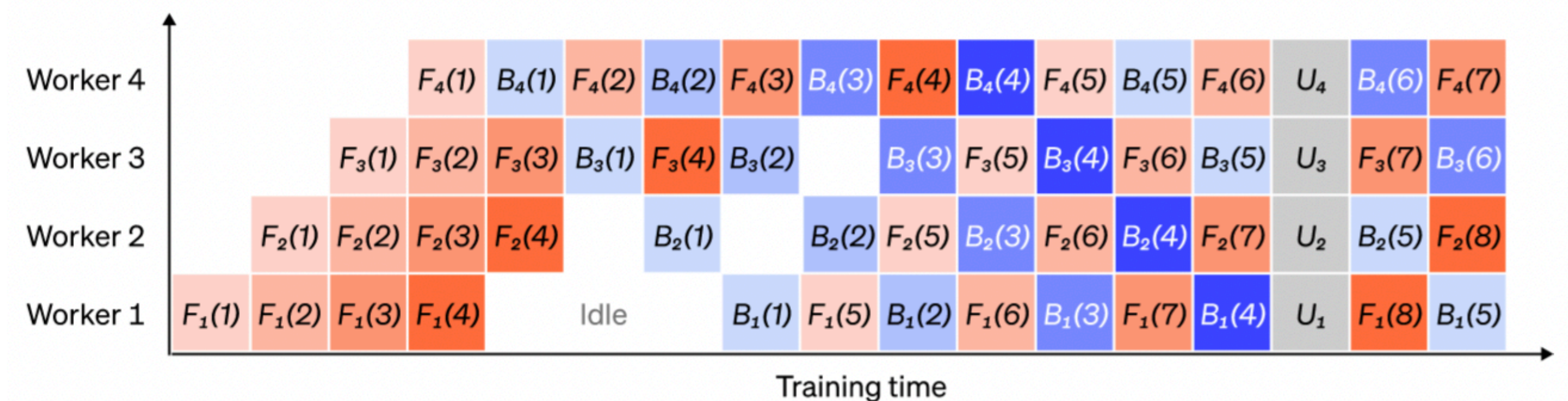


Illustration of a naive pipeline parallelism setup where the model is vertically split into 4 partitions by layer. Worker 1 hosts model parameters of the first layer of the network (closest to the input), while worker 4 hosts layer 4 (which is closest to the output). "F", "B", and "U" represent forward, backward and update operations, respectively. The subscripts indicate on which worker an operation runs. Data is processed by one worker at a time due to the sequential dependency, leading to large "bubbles" of idle time.

## PipeDream



<https://openai.com/research/techniques-for-training-large-neural-networks>

Dorothea Sommer | GWDG | 7.11.2023

# Tensor-parallelism

- Idea: think of matrix multiplication as dot-product between pairs of rows and columns, so it can be splitted among GPUs
- Example: Megatron from Nvidia for Transformers



# Q&A

- **Course certificates:** If you need a printed certificate for course participation, please write an e-mail to [dorothea.sommer@gwdg.de](mailto:dorothea.sommer@gwdg.de)
- **Your course accounts can be used until 8.11.2023, 18.00!**
- Who can get **access to Emmy**, specifically also for projects?  
<https://pad.gwdg.de/s/cAA-M2vpl#>
- **Other questions:** Is there anything you would
  - like to discuss regarding deep learning/GPU?
  - see covered in another course?