

Good Programming Practices, Computer Tools and QOL Tricks

...

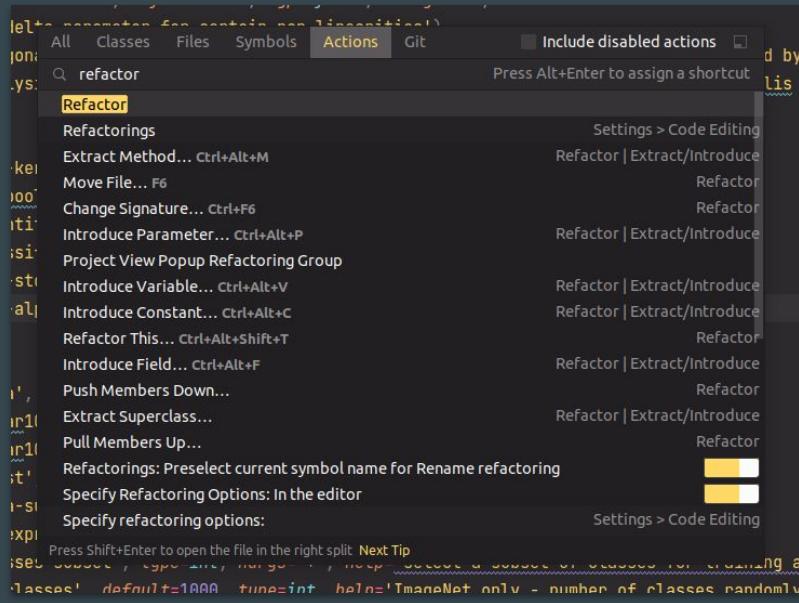
Florentin Guth
CSD PhD Seminar - December 6, 2021

Outline

- **Editing**
- **Programming**
- **Python Tricks**
- **Debugging**
- **Git and GitHub**
- **Visualization**
- **SSH**
- **Writing**

Editing

Editor - PyCharm



- Powerful and easy to use
- Ctrl-Shift-A: universal key binding
- Refactoring tools
- Code-completion and inspection
 - Help it with type annotations

```
import numpy as np

def softmax(x: np.ndarray):
    x_.
        shape
        item(self, args)
        view(self, dtype, args, kwargs)
        std(self, axis, dtype, out, ddof, keepdims, args, ...)
        mean(self, axis, dtype, out, keepdims, args, kwargs)
        all(self, axis, out, keepdims, args, kwargs)
        any(self, axis, out, keepdims, args, kwargs)
        argmax(self, axis, out)
        argmin(self, axis, out)
        argpartition(self, kth, axis, kind, order)
        argsort(self, axis, kind, order)
        astype(self, dtype, order, casting, subok, copy)
```

Get paid version for free with
student email address!

Editor - Vim

- Command-line editor (useful for ssh or quick edits)
- Lots of commands to edit text quickly
- Steep learning curve but worth it
- Possible to use vim keybindings in PyCharm, your browser, Overleaf...



```
florentin@lappy:~/.../Thèse/Random lunches$ vim main.py
```

Usually pre-installed!

```
def get_cost(configuration, participants, history):
    """ Measure the cost of a configuration according to past history of random lunches.
    :param configuration: list of sorted lists of integers, corresponding to participant indices.
    :param participants: DataFrame sorted by email addresses.
    :param history: dictionary of sorted email address pairs to number of past lunches.
    :return: the cost of the configuration (an integer), the smaller the better.
    """
    cost = 0
    for pair in enumerate_pairs(configuration):
        p0, p1 = participants.iloc[pair[0]], participants.iloc[pair[1]]
        emails = str(tuple(sorted((p0[email_field], p1[email_field]))))
        num_times = history.get(emails, 0)
        if p0[team_field] == p1[team_field]:
            num_times += 1000
        cost += num_times ** 2
    return cost

def update_history(configuration, participants, cost, payer_team, history):
    """ Updates in-place the history to account for the chosen configuration. """
    for pair in enumerate_pairs(configuration):
        # Record the pair in the history.
        emails = str(tuple(sorted((participants.loc[participants.index[pair[0]], email_field],
                                    participants.loc[participants.index[pair[1]], email_field]))))
        history[emails] = cost
```

Programming

Commenting and Documenting

Start by explaining what your function, object, code does, and how it does it.



```
def complex_conv2d(x, w):
    """ (B, C, M, N) complex and (K, C, H, W) complex to (B, K, M', N') complex """
    x = complex_to_real_channels(x, separate_real_imag=True) # (B, 2C, M, N)
    w = torch.cat([torch.cat([w.real, -w.imag], dim=1), torch.cat([w.imag, w.real], dim=1)], dim=0) # (2K, 2C, H, W)
    y = F.conv2d(x, w) # (B, 2K, M', N')
    return real_to_complex_channels(y, separate_real_imag=True) # (B, K, M', N') complex
```

```
def get_group_sizes(num_participants):
    """ Break the number of participants into group sizes.
    We try to make groups that are of size ideal_group_size or ideal_group_size +/- 1.
    :return: a dictionary whose keys are group sizes and whose values are the number of such groups.
    """
    # Start with k groups of size group_size and a smaller rest.
    num_full_groups = num_participants // ideal_group_size
    rest = num_participants % ideal_group_size

    # If there are not enough participants, then we can only make one group.
    if num_full_groups == 0:
        groups = {rest: 1}

    # If the rest is empty, we are done, there are only groups of the ideal size.
    elif rest == 0:
        groups = {ideal_group_size: num_full_groups}

    # Otherwise, we try to make groups of size ideal_group_size - 1 or ideal_group_size + 1, if possible.
    # This is done by moving participants from full groups to the rest groups, or by moving participants from the rest
    # group to full groups.
    else:
```

Write explicitly all necessary information to understand or check your code (assumptions, shapes of tensors, ...).

Give explicit names to variables and functions, even if they are long.

```
def get_random_minimal_cost_configuration(participants, group_sizes, history):
    """ Returns a random configuration among those of minimal cost. """
```

Factorization

- Copy-pasting code is usually harmful in the long run
- Factorize (compress) away redundant code
- Introduce pertinent variables, use (local) functions, loops...
- Dictionaries provide a nice switch statement

```
def dataset(train: bool): # Returns the train or validation dataset
    dataset_class = dict(CIFAR10=datasets.CIFAR10, CIFAR100=datasets.CIFAR100,
                         MNIST=datasets.MNIST, ImageNet=datasets.ImageFolder)[dataset_name]
    kwargs = dict(
        root=os.path.join(args.data, "train" if train else "val") if dataset_name == "ImageNet" else "./data",
        transform=transforms.Compose([train_transforms if train else test_transforms] + common_transforms))
    if dataset_name != "ImageNet":
        kwargs.update(train=train, download=True)
    return dataset_class(**kwargs)

print_and_write(f"Working on {dataset_name}", logfile, summaryfile)
train_dataset = dataset(train=True)
val_dataset = dataset(train=False)
```

```
if args.cifar10:
    print_and_write("Working on CIFAR10", logfile, summaryfile)

if args.grayscale:
    train_dataset = datasets.CIFAR10(
        root='./data', train=True, download=True, transform=transforms.Compose([
            transforms.RandomHorizontalFlip(),
            transforms.RandomCrop(32, padding=4),
            transforms.Grayscale(),
            transforms.ToTensor(),
            normalize,
        ]))

    val_dataset = datasets.CIFAR10(
        root='./data', train=False, download=True, transform=transforms.Compose([
            transforms.Grayscale(), transforms.ToTensor(), normalize]))

else:
    train_dataset = datasets.CIFAR10(
        root='./data', train=True, download=True, transform=transforms.Compose([
            transforms.RandomHorizontalFlip(),
            transforms.RandomCrop(32, padding=4),
            transforms.ToTensor(),
            normalize,
        ]))

    val_dataset = datasets.CIFAR10(
        root='./data', train=False, download=True, transform=transforms.Compose([
            transforms.ToTensor(), normalize]))

elif args.cifar100:
    print_and_write("Working on CIFAR100", logfile, summaryfile)

    train_dataset = datasets.CIFAR100(root='./data', train=True,
                                      download=True, transform=transforms.Compose([
                                          transforms.RandomHorizontalFlip(),
                                          transforms.RandomCrop(32, padding=4),
                                          transforms.ToTensor(),
                                          normalize,
                                      ]))

    val_dataset = datasets.CIFAR100(root='./data', train=False, download=True,
                                    transform=transforms.Compose([transforms.ToTensor(), normalize]))
```

Architecture Design

```
def build_layers(input_type: SplitTensorType, modules, i, args):
    """ Builds a list of layers from a list of modules.
    :param input_type:
    :param modules: list of strings describing the layers
    :param i: index of the current block
    :param args: command-line arguments
    :return: Sequential module
    """
    builder = Builder(input_type)

    def branching_kwargs(**submodules): ...

    for module in modules:
        if module == "Fw": ...

        elif module == "R":
            builder.add_batched(Realifier)

        elif module == "C":
            builder.add_batched(Complexifier)

        elif module == "B": ...

        elif module in ["mod", "rho"]:
            ...

        elif module == "Std":
            builder.add_batched(Standardization)

        elif module in ["P", "Pr", "Pc"]:

    return builder.build()

def main():
    parser = ArgumentParser()
    parser.add_argument("--arch", type=str, required=True,
                        help="Architecture string consisting of layers separated by spaces.")
```

- It can be useful to sit and think about how to implement a new feature (consider several options)
- Think from the point of view of the user: what would I like to be able to write?
- Plan for the future: more general and flexible is usually better
- Saves a lot of time if everything is customizable from command-line arguments
- Monolithic functions or objects are harder to modify

```
python main.py --arch "Fw rho Std P"
python main.py --arch "Fw rho P"
python main.py --arch "Fw mod Std R C"
python main.py --arch "Fw rho P mod"
```

Python Tricks

String Manipulation

eval can be used to allow powerful command-line customization.

```
In [3]: x = np.random.random(10)
In [4]: f'{x.shape} {x.min():.2f} {x.max():.2f}'
Out[4]: '(10,) x.min()=0.06936024503854088 x.max()=0.87'
```

f-strings are the ultimate logging tool.

```
class Branching(nn.Module):
    """ Implements a branching path. Expects a dict of split tensors, applies a submodule to each one
    and merges the result. """
    def __init__(self, input_type: Dict[str, SplitTensorType], **kwargs):
        """
        :param input_type: dictionary of SplitTensorTypes, for each branch of the path
        :param kwargs: of the form `key`_module_class and `key`_module_kwargs
        """
        super().__init__()

        self.input_type = input_type

        submodules = {}
        for key, input_type in self.input_type.items():
            module_class = kwargs.pop(f"{key}_module_class", Identity)
            module_kwargs = kwargs.pop(f"{key}_module_kwargs", {})
            submodules[key] = module_class(input_type, **module_kwargs)
        self.submodules = ModuleDict(submodules)
        assert len(kwargs) == 0
```

```
if args.data_subset is not None:
    train_dataset = torch.utils.data.Subset(train_dataset, eval(args.data_subset))
```

Dictionaries provide a way to manipulate variables by name (keyword arguments, locals).

Use sparingly!

```
def compute_many_things():
    some_var = ...
    some_other_var = ...
    another_one = ...
    do_you_get_the_point = ...
    return locals()
```

```
In [5]: for i in range(10):
...:     locals()[f"var{i}"] = i
...:
In [6]: var5
Out[6]: 5
```

Capture Sub-Expressions with the Walrus Operator

The walrus operator (:=) allows simplifying the code in many cases.

```
scores = [score for item in iterable if (score := get_score(item)) is not None]
```

```
def argmax(iterable, score):
    """ Finds the first item in iterable with the highest score. """
    best_item = None
    best_score = float("-inf")
    for item in iterable:
        if (item_score := score(item)) > best_score:
            best_item = item
            best_score = item_score
    return best_item
```

```
if any(is_problematic(witness := item) for item in iterable):
    raise ValueError(f"Problem with {witness}")
```

Dealing with Long Computations

```
def get_random_minimal_cost_configuration(participants, group_sizes, history):
    """ Returns a random configuration among those of minimal cost. """
    num_participants = len(participants)

    # Compute all configuration of with minimal cost.
    best_cost = float("inf")
    best_configurations = []
    try:
        iterator = tqdm.tqdm(enumerate_random_configurations(num_participants, group_sizes),
                             total=num_configurations(num_participants, group_sizes))
        for configuration in iterator:
            # Translate participants from numbers to email addresses.
            cost = get_cost(configuration, participants, history)

            if cost < best_cost:
                best_cost = cost
                best_configurations = [configuration]
            elif cost == best_cost:
                best_configurations.append(configuration)
        iterator.desc = f"Found {len(best_configurations)} configurations of cost {best_cost}"
    except KeyboardInterrupt:
        # Impatient user, just return one of the best configurations found so far
        pass

    # Return a random configuration among all best ones.
    return best_configurations[random.randrange(0, len(best_configurations))]
```

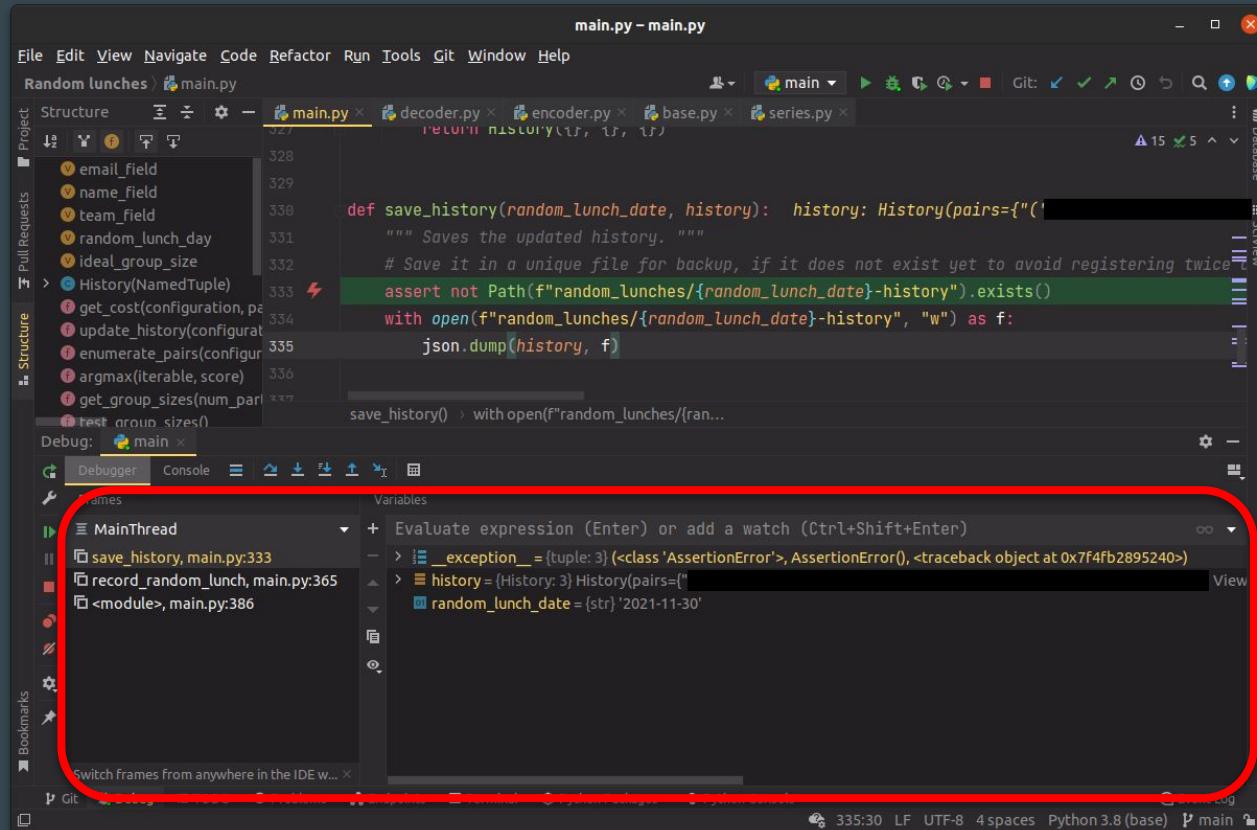
- Use tqdm to easily have a running time estimate
- Catch KeyboardInterrupt to perform final operations even when interrupted
- Cache results to disk using json or pickle to avoid redoing slow computations

```
filename = self.full_path(name, suffix=".pt", level=level)
if Path(filename).exists():
    return torch.load(filename)
ret = closure(*args, **kwargs)
torch.save(ret, filename)
```

```
(base) florentin@lappy:~/.../Thèse/Random lunches$ python main.py
Found 1 configurations of cost 4000003:  0% | 5647/4509264634875.0 [00:19<4340128:20:29, 288.60it/s]
```

Debugging

Visual Debugging with PyCharm



The screenshot shows the PyCharm IDE interface. The main window displays a Python file named `main.py` with the following code:

```
def save_history(random_lunch_date, history):    history: History(pairs={})  
    """ Saves the updated history. """  
    # Save it in a unique file for backup, if it does not exist yet to avoid registering twice  
    assert not Path(f"random_lunches/{random_lunch_date}-history").exists()  
    with open(f"random_lunches/{random_lunch_date}-history", "w") as f:  
        json.dump(history, f)  
  
save_history() > with open(f"random_lunches/{ran...")
```

The code has a breakpoint at line 333. A red box highlights the `Variables` tool window, which shows the current state of variables:

- `_exception_ = (tuple: 3) (<class 'AssertionError'>, AssertionError, <traceback object at 0x7f4fb2895240>)`
- `history = {History: 3} History(pairs={})`
- `random_lunch_date = {str} '2021-11-30'`



- Interrupt on error (or manually)
- Trace execution
- Evaluate expressions
- Run step by step

Preventive Debugging

Do sanity checks with assertions or explicit raises (much better than silent errors).

```
▶ def test_enumerate_groups():
    """ Small function to test enumeration of all possible groups. """
    from scipy.special import binom
    num_participants = 4
    for group_size in range(num_participants + 2):
        groups = list(enumerate_groups(num_participants, group_size))
        assert all(len(group) == group_size for group in groups)
        assert len(groups) == binom(num_participants, group_size)
    print(num_participants, group_size, groups)
```

Log relevant variables for post-execution checks.

```
logger.info(f"Command line: {' '.join(sys.argv)}")
```

```
if args.remove_L[i]:
    pass
elif i < len(args.separation_sizes):
    assert False # TODO: this is deprecated
    total_phases, linear_phases = scattering_non_lin.phases()
    proj = FwStructuredProj(
        nrev_out_channels=around_l= args.scat_angles[i], A=4
```

Test early and independently if feasible.

Rubber Duck Debugging

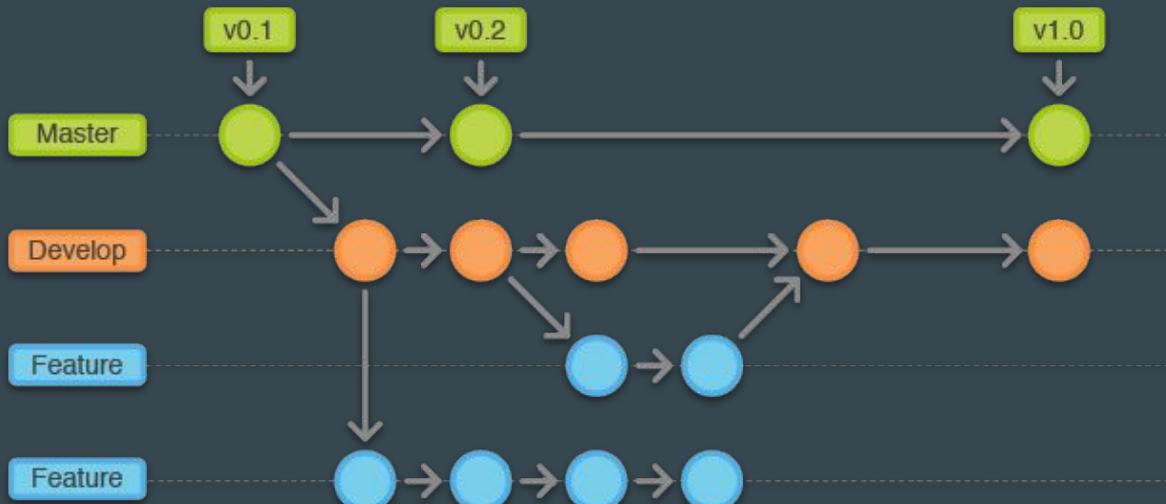
If all hope is lost: explain your problem in detail to your rubber duck (or your exasperated officemate). Explain how your program actually behaves and why it behaves this way. This usually helps :)



Git and GitHub

Git

- Track your code changes
- Rollback to a previous version of your code
- Have multiple current versions with branches
- Know when a bug was introduced
- Easier to use with PyCharm integration



GitHub



GitHub

Commit Changes

Changelist: Changes Git

Author: FlorentinGuth

Amend commit Sign-off commit

Before Commit Reformat code

Deleted unused function 'enumerate_configurations'.

1 main.py

Commit Message

Deleted unused function 'enumerate_configurations'.

Diff

Side-by-side viewer Do not ignore Highlight words

Your version

1 difference

```
11de2370b0a22e6d1a79ac887477de7a06ffe71
assert len(groups) == binom(num_participants, group_size)
print(num_participants, group_size, groups)

def enumerate_configurations(num_participants, group_size):
    """ Returns an iterator over all possible configurations.
    Participants are represented here as an integer f.
    :return: iterator over configurations, which are tuples of integers.
    """
    if num_participants == 0:
        yield []
    else:
        # We start by enumerating possible groups with size 1.
        for group_size_0 in group_sizes:
            # Don't include zero in a group of size 1.
            if group_size_0 > 1:
                for group_size_1 in group_sizes[group_size_0 - 1:]:
                    # Don't include zero in a group of size 2.
                    if group_size_1 > 1:
                        for group_size_2 in group_sizes[group_size_1 - 1:]:
                            # Don't include zero in a group of size 3.
                            if group_size_2 > 1:
                                for group_size_3 in group_sizes[group_size_2 - 1:]:
                                    # Don't include zero in a group of size 4.
                                    if group_size_3 > 1:
                                        for group_size_4 in group_sizes[group_size_3 - 1:]:
                                            # Don't include zero in a group of size 5.
                                            if group_size_4 > 1:
                                                for group_size_5 in group_sizes[group_size_4 - 1:]:
                                                    # Don't include zero in a group of size 6.
                                                    if group_size_5 > 1:
                                                        for group_size_6 in group_sizes[group_size_5 - 1:]:
                                                            # Don't include zero in a group of size 7.
                                                            if group_size_6 > 1:
                                                                for group_size_7 in group_sizes[group_size_6 - 1:]:
                                                                    # Don't include zero in a group of size 8.
                                                                    if group_size_7 > 1:
                                                                        for group_size_8 in group_sizes[group_size_7 - 1:]:
                                                                            # Don't include zero in a group of size 9.
                                                                            if group_size_8 > 1:
                                                                                for group_size_9 in group_sizes[group_size_8 - 1:]:
                                                                                    # Don't include zero in a group of size 10.
                                                                                    if group_size_9 > 1:
                                                                                        for group_size_10 in group_sizes[group_size_9 - 1:]:
                                                                                            # Don't include zero in a group of size 11.
                                                                                            if group_size_10 > 1:
                                                                                                for group_size_11 in group_sizes[group_size_10 - 1:]:
                                                                                                 ...
```

170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195

Commit Cancel

- Collaborate with other people on code
- Sync your code across computers
- Easy online backup
- Free pro account with student email address
- Easier to use with PyCharm integration

FlorentinGuth/RandomLunch

github.com/FlorentinGuth/RandomLunch

Actus Thèse Administratif Projets Content Misc Temp wallpapers Recettes Musique

Pulls Issues Marketplace Explore

FlorentinGuth / RandomLunch Public

Code Issues Pull requests Actions Projects Wiki Security Insights

main Go to file Add file Code About

FlorentinGuth Custom group sizes, random enumeration of config... 6 days ago 4

.gitignore Overhauled script usage, with two phases: "prop..." 20 days ago

main.py Custom group sizes, random enumeration of config... 6 days ago

Help people interested in this repository understand your project by adding a README. Add a README

Releases No releases published Create a new release

Packages No packages published Publish your first package

Languages Python 100.0%

Visualization

Tensorboard



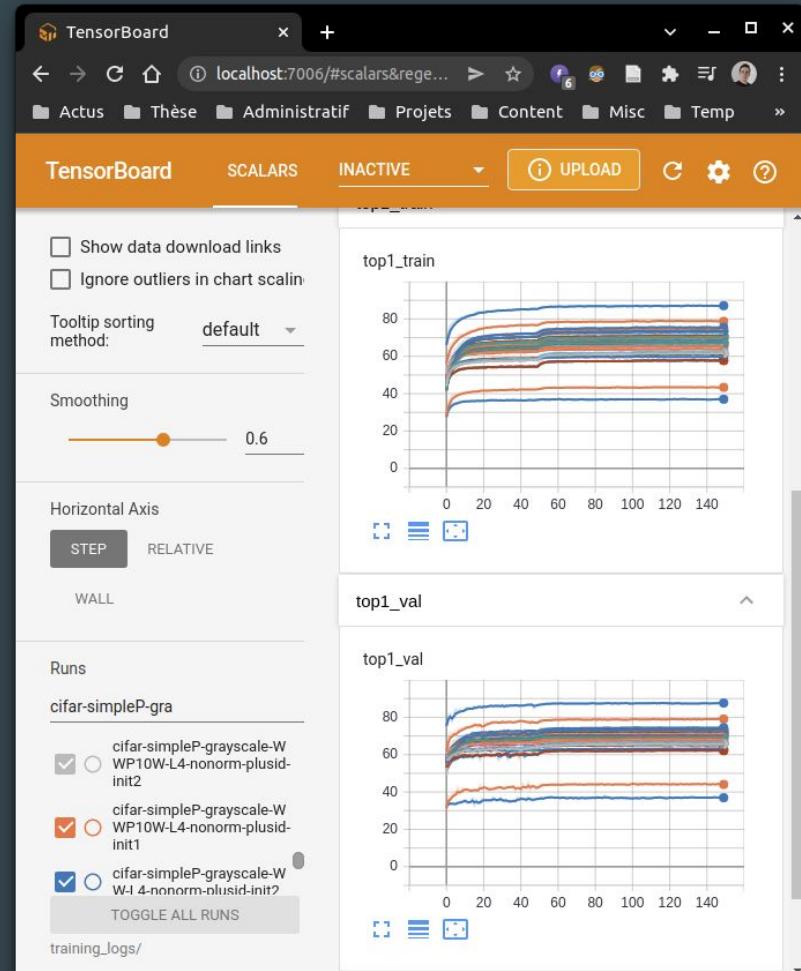
TensorBoard

- Monitor running experiments
- Compare them to previous ones
- Visualize the evolution of scalars, images...
- Easy-to-use web interface

```
from torch.utils.tensorboard import SummaryWriter  
writer = SummaryWriter(logs_dir)
```

```
suffix = "train" if is_training else "val"  
writer.add_scalar(f'top5_{suffix}', top5.avg, global_step=epoch)  
writer.add_scalar(f'top1_{suffix}', top1.avg, global_step=epoch)
```

```
(torchjz) fguth@rusty1:~/sparse_scat_net$ tensorboard --logdir training_logs/  
TensorFlow installation not found - running with reduced feature set.  
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all  
TensorBoard 2.3.0 at http://localhost:6006/ (Press CTRL+C to quit)
```



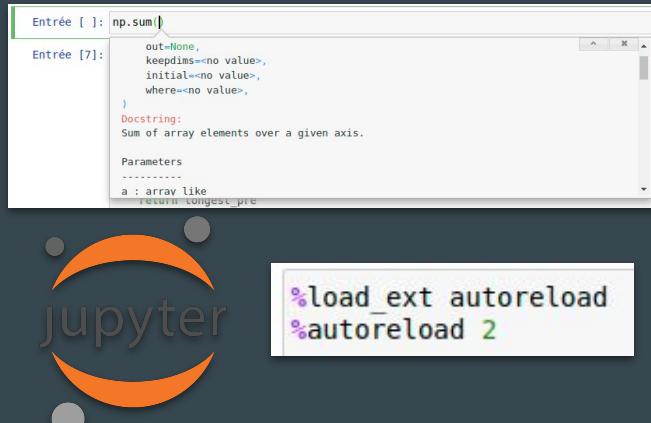
Jupyter Notebooks

A screenshot of a Jupyter Notebook interface. The title bar shows "projectors_analysis - Jupyter". The URL in the address bar is "localhost:9888/notebooks/projectors_analysis.ipynb". The notebook content includes a section titled "Lucent Visualization" with the subtitle "Images which activate the most a given channel." Below this, a code cell (Entrée [79]) contains Python code to generate a grid of 10x10 grayscale images. The output of this cell is a 4x10 grid of images showing various patterns, likely visualizations of neural network activations.

```
(base) florentin@lappy:~/.../Classification/code$ jupyter-notebook
[I 2021-12-04 20:12:00.717 LabApp] JupyterLab extension loaded from /home/florentin/anaconda3/lib/python3.8/site-packages/jupyterlab
[I 2021-12-04 20:12:00.718 LabApp] JupyterLab application directory is /home/florentin/anaconda3/share/jupyter/lab
[I 20:12:00.721 NotebookApp] Serving notebooks from local directory: /home/florentin/Documents/Thèse/Classification/code
[I 20:12:00.721 NotebookApp] Jupyter Notebook 6.4.3 is running at:
[I 20:12:00.721 NotebookApp] http://localhost:8888/?token=cd37947d7d88bf263bbb0cc683ac7463813012e77e5a2d08
```

```
Entrée [79]: n = len(images)
plt.figure(figsize=(10, n))
i = 0
for imgs in images:
    for img in imgs[0]:
        i += 1
        plt.subplot(n, 10, i)
        plt.imshow(img, cmap="gray")
        plt.axis("off")
plt.tight_layout()
plt.show()
```

- Cells can contain python code or markdown/LaTeX
- Show outputs (e.g. plots)
- Shift-Tab for doc
- Ctrl-click for multiple cursors
- Best used with other imported files and autoreload



SSH

SSH Configuration

- Set .ssh/config for proxy jumps and port forwardings (run tensorboard and notebooks remotely!)
- Generate a private key for no-password logins

```
Host di
  Hostname di.ens.fr
  User guth
  LocalForward 8888 localhost:8888
  LocalForward 8889 localhost:8889
  LocalForward 6006 localhost:6006
  LocalForward 6007 localhost:6007
  LocalForward 6008 localhost:6008
  LocalForward 6009 localhost:6009

Host genievre
  HostName genievre
  ProxyJump di
  User guth
  LocalForward 8888 localhost:8888
  LocalForward 8889 localhost:8889
  LocalForward 6006 localhost:6006
  LocalForward 6007 localhost:6007
  LocalForward 6008 localhost:6008
  LocalForward 6009 localhost:6009
```

```
(base) florentin@lappy:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/florentin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
(base) florentin@lappy:~$ ssh-copy-id genievre
```

>
—
SSH

Terminal Multiplexer



The screenshot shows a tmux session with three windows. The leftmost window displays two identical tables of GPU usage data from nvidia-smi. The rightmost window shows a terminal multiplexer interface with several tabs open, including one for a file named 'rusty1' and another for 'rusty1:~\$'. The bottom status bar indicates the session ID is 789, the date is Dec 3, and the time is 16:17:59.

GPU	GI	CI	PID	Type	Process name	GPU Memory Usage
0	N/A	N/A	645030	C	.../envs/torchjz/bln/python	3061MB

GPU Name	Persistence-MI	Bus-Id	Disp.A	Volatile Uncorr. ECC	Memory-Usage	GPU-Util Compute M.	MIG M.
0 Tesla V100-SXM2...	On	00000000:0:E:00.0	Off	0	3064MB / 32510MB	96%	Default N/A

GPU	GI	CI	PID	Type	Process name	GPU Memory Usage
0	N/A	N/A	645030	C	.../envs/torchjz/bln/python	3061MB

GPU Name	Persistence-MI	Bus-Id	Disp.A	Volatile Uncorr. ECC	Memory-Usage	GPU-Util Compute M.	MIG M.
0 Tesla V100-SXM2...	On	00000000:0:E:00.0	Off	0	3064MB / 32510MB	94%	Default N/A

GPU	GI	CI	PID	Type	Process name	GPU Memory Usage
0	N/A	N/A	645030	C	.../envs/torchjz/bln/python	3061MB


```
0:2:vim - "rusty1"
781     module_info.append(f'\n - Quadrature on {frame_name}: ratio norm W MT / norm W2 ({norm_ratio:.3f})')
782     module_info.append(f'\n - Frame norm of {frame_name}: ({norm_frame:.3f})')
783
784     if len(module_info) > 1:
785         module_info.extend([f'\n- {module_name} ({module.__class__.__name__}):'] + module_in
786
787     print_and_write(''.join(module_info), logfile, summaryfile)
788
789
790 def one_epoch(loader, model, criterion, optimizer, epoch, args, logfile, summaryfile, writer, is_
791 training):
792     batch_time = AverageMeter('Time', ':.1f')
793     data_time = AverageMeter('Data', ':.1f')
794     losses = AverageMeter('Loss', ':.4f')
795     top1 = AverageMeter('Acc@1', ':.1f')
796     top5 = AverageMeter('Acc@5', ':.1f')
797     name_epoch = "Train" if is_training else "Validation"
798     progress = ProgressMeter(
799         len(loader), [batch_time, data_time, losses, top1, top5],
800         prefix=() if is_training else f'model.train()')
801
802     if is_training:
803         model.train()
804     else:
805         model.eval()
806
807     NORMAL +0 -0 -0 ! master! main_block.py[+]
808     python utf-8[unix] 76% ≡ 789/1030 ln : 1
809
810
811 (torchjz) fguth@rusty1:~/sparse_scat_net$ sq
812          208ID PARTITION      NAME    USER ST   TIME NODES NODELIST(REASON)
813          119c199    gpu    bash fguth R 1:09:54:03      1 workergpu053
814
815 (torchjz) fguth@rusty1:~/sparse_scat_net$
```

- Persistent processes across logouts
- Several terminals in one session

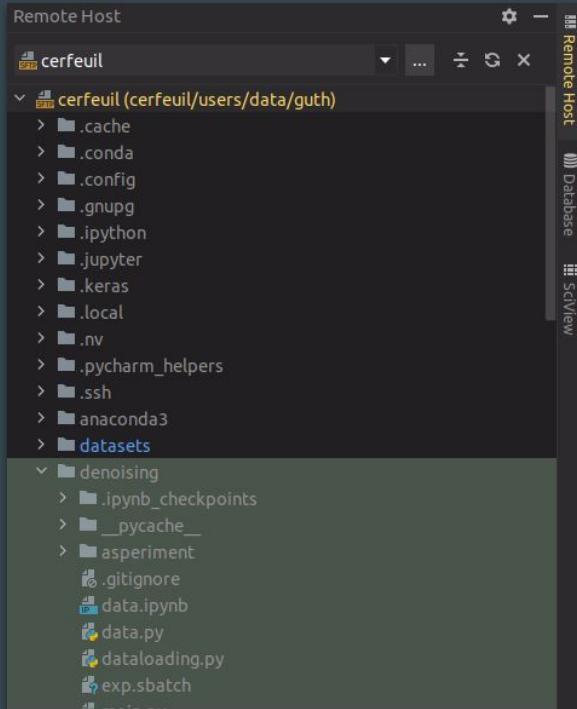
Other Bash Tools

- Aliases in .bashrc
- Search bash history with Ctrl-R
- Type man + command to access its help
- Browse files with less (or vim)
- Lot of useful commands: grep, pipes, redirections...
- Script in python rather than bash!

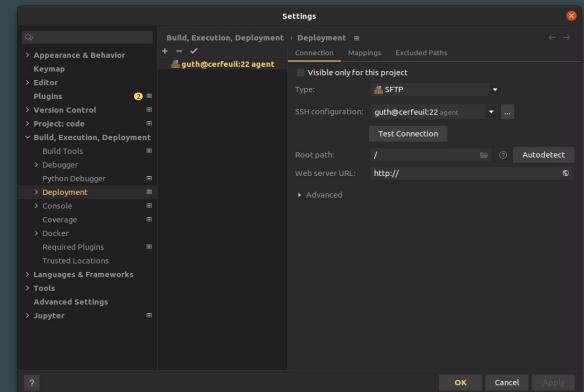
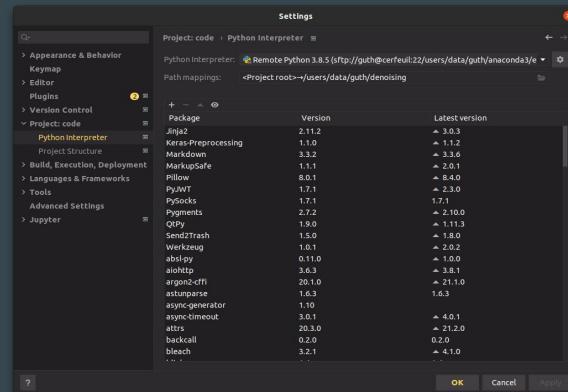


```
(reverse-i-search)`tens': tensorboard --logdir training_logs/
126 #Aliases
127 alias reload='source ~/.bashrc'
128 alias ll='ls -lrth'
129 alias ca='conda activate'
130 alias cda='conda deactivate'
131 alias tb='tensorboard --logdir'
132 alias sq='squeue -u fguth'
133 alias sra100='srun -N 1 -c 10 --mem 100G -p gpu --gpus=a100-40gb:1 --constraint=a100 --constraint=a100-40gb --pty bash -i'
134 alias srv100='srun -N 1 -c 10 --mem 100G -p gpu --gpus=v100-32gb:1 --constraint=v100 --constraint=v100-32gb --pty bash -i'
135 alias port='ssh -L 9888:localhost:9888' # usage: port workergpuN, after srun and in another pane
```

Remote Editing and Uploading/Downloading with PyCharm

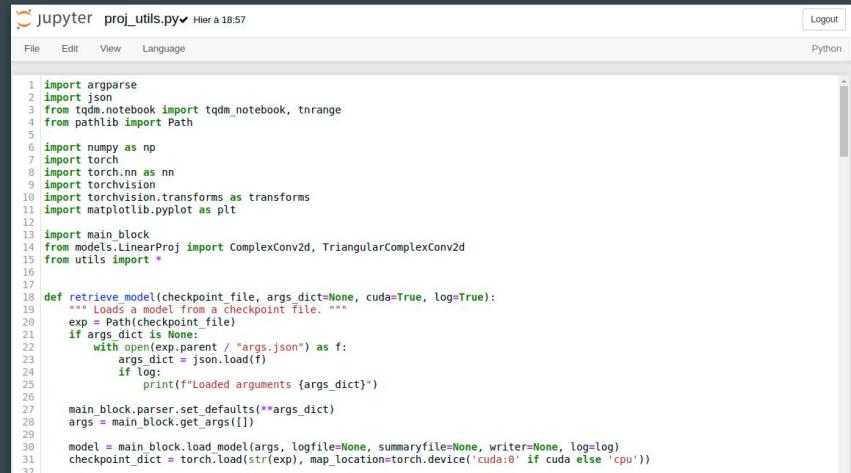


- Automatically upload files
- Remote execution and debugging
- Show remote files in the editor, download them easily



Remote Editing and Uploading/Downloading with Other Tools

- Jupyter can edit python files and produced images can be directly saved locally
- GitHub can be used to sync files
- Vim for remote editing,
rsync for upload/download



```
1 import argparse
2 import json
3 from tqdm.notebook import tqdm_notebook, tnrange
4 from pathlib import Path
5
6 import numpy as np
7 import torch
8 import torch.nn as nn
9 import torchvision
10 import torchvision.transforms as transforms
11 import matplotlib.pyplot as plt
12
13 import main.block
14 from models.LinearProj import ComplexConv2d, TriangularComplexConv2d
15 from utils import *
16
17
18 def retrieve_model(checkpoint_file, args_dict=None, cuda=True, log=True):
19     """ Loads a model from a checkpoint file. """
20     exp = Path(checkpoint_file)
21     if args_dict is None:
22         with open(exp.parent / "args.json") as f:
23             args_dict = json.load(f)
24         if log:
25             print(f"Loaded arguments {args_dict}")
26
27     main.block.parser.set_defaults(**args_dict)
28     args = main.block.get_args()
29
30     model = main.block.load_model(args, logfile=None, summaryfile=None, writer=None, log=log)
31     checkpoint_dict = torch.load(str(exp), map_location=torch.device('cuda:0' if cuda else 'cpu'))
```

A screenshot of a Jupyter Notebook interface titled "Jupyter proj_utils.py". The code in the cell is for retrieving a model from a checkpoint file. It uses argparse to handle command-line arguments, json to load configuration files, and various PyTorch and torchvision modules for model loading and processing. The code includes comments explaining its purpose and how it handles arguments and logs.

Writing

LaTeX Tips

LaTeX

Consider creating your own preamble file with general packages and useful commands. Commands can be shorthands for common sub-expressions or a way to remember the syntax for something.

```
\usepackage{subdepth}
\let\originalleft\left
\let\originalright\right
\renewcommand{\left}{\mathopen{}\mathclose{}\bgroup\originalleft}
\renewcommand{\right}{\aftergroup\egroup\originalright}
```

Use this to correct maths typesetting (fix a bug in left/right, handles indices and exponents properly).



The Detexify website/app gives the command corresponding to the hand-drawn symbol.

```
\newcommand{\RR}{\mathbb{R}}
\newcommand{\CC}{\mathbb{C}}
\newcommand{\NN}{\mathbb{N}}
\newcommand{\ZZ}{\mathbb{Z}}
\newcommand{\QQ}{\mathbb{Q}}
\newcommand{\KK}{\mathbb{K}}
\newcommand{\EE}{\mathbb{E}}
\newcommand{\PP}{\mathbb{P}}
\newcommand{\paren}[1]{\left( #1 \right)}
\newcommand{\parenn}[1]{\left( \left. #1 \right. \right)}
\newcommand{\bracket}[1]{\left[ \left. #1 \right. \right]}
\newcommand{\brackett}[1]{\left[ \left[ #1 \right] \right]}
\newcommand{\curly}[1]{\left\{ \left. #1 \right. \right\}}
\newcommand{\curlyy}[1]{\left\{ \left[ #1 \right] \right\}}
\newcommand{\intint}[1]{\left\langle \left. #1 \right. \right\rangle}
\newcommand{\intintt}[1]{\left\langle \left[ \left. #1 \right. \right] \right\rangle}
\newcommand{\norm}[1]{\left\| \left. #1 \right. \right\|_{\mathrm{VVert}}}
\newcommand{\normm}[1]{\left\| \left[ \left. #1 \right. \right] \right\|_{\mathrm{VVert}}}
\newcommand{\abs}[1]{\left| \left. #1 \right. \right|_{\mathrm{lvert}\, \mathrm{rvert}}}
\newcommand{\abss}[1]{\left| \left[ \left. #1 \right. \right] \right|_{\mathrm{lvert}\, \mathrm{rvert}}}
\newcommand{\inner}[1]{\left\langle \left. \langle \right. \right\rangle}
\newcommand{\innerr}[1]{\left\langle \left[ \left. \langle \right. \right] \right\rangle}
\newcommand{\floor}[1]{\left\lfloor \left. #1 \right. \right\rfloor_{\mathrm{lfloor}\, \mathrm{rfloor}}}
\newcommand{\floorr}[1]{\left\lfloor \left[ \left. #1 \right. \right] \right\rfloor_{\mathrm{lfloor}\, \mathrm{rfloor}}}
\newcommand{\expect}[1]{\left\langle \left. \langle \right. \right\rangle_{\mathrm{EE}\, \mathrm{bracket}\{#1\}}}
\newcommand{\expectt}[1]{\left\langle \left[ \left. \langle \right. \right] \right\rangle_{\mathrm{EE}\, \mathrm{brackett}\{#1\}}}
\newcommand{\outerprod}[2]{\left\langle \left. \#1 \middle\#2 \right. \right\rangle_{\mathrm{outerprod}}}
\newcommand{\stt}[2]{\left\langle \left. \#1 \middle\#2 \right. \right\rangle_{\mathrm{stt}}}
\newcommand{\trans}[1]{\mathrm{trans}^{\{ \mathrm{mathrm}\, T \}}\{#1\}^{-1}}
\newcommand{\inv}[1]{\mathrm{inv}\{#1\}^{\{ \mathrm{#1} \}^{-1}}}
\newcommand{\ind}[1]{\mathrm{ind}\{#1\}_{\mathrm{mathds}\{#1\}_{\{ \mathrm{#1} \}}}}
\newcommand{\diff}[1]{\mathrm{diff}(\mathrm{mathrm}\{d\})\{#1\}}
\newcommand{\summ}[2]{\mathrm{sum}\{ \mathrm{#1} = 1 \}^{\{ \mathrm{#2} \}}}
\newcommand{\sums}[2]{\mathrm{sums}\{ \mathrm{#1} = 1 \}^{\{ \mathrm{#2} \}}}
\newcommand{\prodd}[2]{\mathrm{prod}\{ \mathrm{#1} = 1 \}^{\{ \mathrm{#2} \}}}
\newcommand{\prodts}[2]{\mathrm{prod}\{ \mathrm{#1} = 1 \}^{\{ \mathrm{#2} \}}_{\mathrm{times}\, \mathrm{cdots}\, \mathrm{times}}}
\newcommand{\extt}[2]{\mathrm{ext}\{#2\}_{\mathrm{bigwedge}\{ \mathrm{#1} : 1 \}^{\{ \mathrm{#2} \}}}}
\newcommand{\exts}[2]{\mathrm{exts}\{ \mathrm{wedge}\}_{\mathrm{dots}\, \mathrm{wedge}}\{#2\}}
\DeclareMathOperator*{\argmax}{argmax}\{arg\}, max
\DeclareMathOperator*{\argmin}{argmin}\{arg\}, min
\DeclareMathOperator*{\sgn}{sgn}\{sgn\}
\DeclareMathOperator{\Tr}{Tr}\{Tr\}
\DeclareMathOperator{\tr}{tr}\{tr\}
\DeclareMathOperator*{\Vect}{Vect}\{Vect\}
\DeclareMathOperator*{\vect}{vect}\{vect\}
\DeclareMathOperator*{\Span}{Span}\{Span\}
\DeclareMathOperator*{\spann}{span}\{span\}
\DeclareMathOperator*{\Ker}{ker}\{ker\}
\DeclareMathOperator*{\Imm}{Im}\{Im\}
```

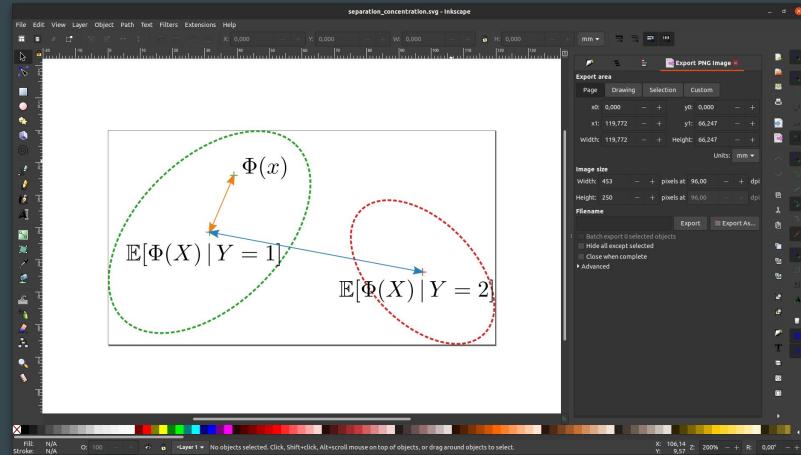
Generating High-Quality Tables and Figures

The booktabs package produce pretty tables (using only horizontal rules is recommended).

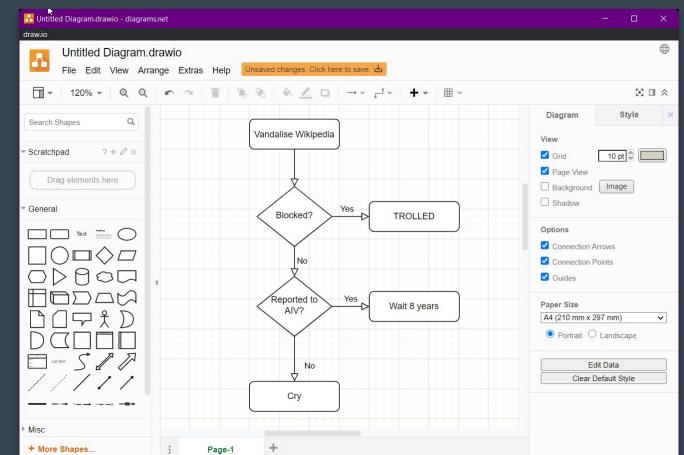
$\Phi(x)$	x	$F^T \rho F x$			$S_T(x)$
		$\rho = \rho_a$	$\rho = \rho_t$	$\rho = \rho_{rt}$	
MNIST	Error (%)	7.4	1.3	1.4	1.3
	Fisher	19	68	69	130
CIFAR	Error (%)	60.5	28.1	34.8	26.5
	Fisher	6.7	15	13	12

```
matplotlib.rcParams.update({'text.usetex': True, 'font.family': 'serif'}) # 'sans-serif' for Beamer
plt.savefig("figure.pdf", transparent=True, bbox_inches="tight", pad_inches=0)
```

Generate pdf output with matplotlib, use LaTeX fonts for axe labels and legends.



Create pdf
figures with
diagrams.net
or Inkscape.



Taking Notes with Typora

Markdown editors like Typora are useful for daily notes. They support rich text: table of contents, images, LaTeX, tables, code...



Untitled - Typora

File Edit Paragraph Format View Themes Help

Files **Outline**

Daily Notes
12/06: Demo

Daily Notes

12/06: Demo

Supports *formatted text*

- Write bullets
 - Arbitrary nested levels
- Write inline math $\sqrt{\sum_{i=1}^n x_i^2}$ or display math:
$$\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = 1$$

	cifar-iclr-L4-skip-Pr-nonorm	92.06	92.06	299	Tue Oct 5, 22:01:12	6h 55m 46s
5	cifar-iclr-L4-skip-Pr-norm (expressions supported)	92.04	92.04	299	Tue Oct 5, 22:03:52	6h 58m 26s
6	cifar-iclr-L4-skip-Pc-norm	91.95	91.95	299	Tue Oct 5, 22:15:24	7h 9m 55s
7	cifar-iclr-L4-skip-Pr-norm	91.83	91.83	299	Tue Oct 5, 22:16:28	7h 11m 1s
8	cifar-iclr-L8-skip-Pr-norm	90.86	90.86	194	Tue Oct 5, 22:24:43	7h 18m 24s
9	cifar-iclr-L8-skip-Pc-norm	90.79	90.79	193	Tue Oct 5, 22:22:55	7h 16m 36s
10	cifar-iclr-L8-skip-Pr-norm	90.74	90.74	195	Tue Oct 5, 22:22:50	7h 16m 32s
11	cifar-iclr-L8-skip-Pc-norm	90.68	90.68	191	Tue Oct 5, 22:24:42	7h 18m 22s
12	cifar-iclr-L4-Pr-norm	88.98	88.98	299	Tue Oct 5, 18:11:20	3h 6m 46s
13	cifar-iclr-L4-Pc-norm	88.7	88.7	299	Tue Oct 5, 19:50:14	4h 45m 17s
14	cifar-iclr-L4-Pr-norm	88.27	88.27	299	Tue Oct 5, 18:00:28	2h 55m 57s
15	cifar-iclr-L8-Pr-norm	87.6	87.6	299	Tue Oct 5, 19:18:57	4h 14m 8s
16	cifar-iclr-L4-Pc-norm	87.52	87.52	299	Tue Oct 5, 19:36:23	4h 31m 27s
17	cifar-iclr-L8-Pc-norm	86.99	86.99	299	Tue Oct 5, 22:17:58	7h 12m 28s
18	cifar-iclr-L8-Pr-norm	85.74	85.74	299	Tue Oct 5, 19:15:29	4h 10m 41s
19	cifar-iclr-L8-Pc-norm	85.5	85.5	299	Tue Oct 5, 22:07:14	7h 1m 47s

Write **code fences** with syntax highlighting:

```
def f(x):
    return sum(y ** 2 for y in x)
```

< >

75 Words

Summary

- Invest time in your tools and skills, it's worth it
- Explain things in natural language before coding or debugging
- Strive for generality and simplicity

Thank you!