SWINIR STUDENT TRAINING WORKFLOW

main_train_student

- Entry point
- The principal, it reads the instructions (.json), and make every files work.
- it takes the .json and create a Hiring Manager = model

```
Python

model = define_Model(opt)
model.init_train()
```

- the hiring manager knows the instructions and will start hiring

```
for epoch in range(1000000):
    for i, train_data in enumerate(train_loader):
        # ...
        model.feed_data(train_data)          # Step A: Give the Manager the images
        model.optimize_parameters(current_step) # Step B: Tell Manager to train one step
```

- the principal thread the hiring manager as a "black box", it does not know who the manager's hired worker.

Select_model

- only a switchboard file

```
def define_Model(opt):
    model = opt['model'] # Reads "plain" from your JSON

    if model == 'plain':
        from models.model_plain import ModelPlain as M
    elif model == 'gan':
        # ... (we don't use this)

    m = M(opt) # Creates the class instance
    return m
```

- it connect the instructions (.json) to professor = ModelPlain

model_plain

- The most important logic file.
- it holds the comparison for Students and Teacher
- it feeds them data, calculate "grade" (loss), and updates the student's brain

```
class ModelPlain(ModelBase):
    def __init__(self, opt):
        super(ModelPlain, self).__init__(opt)
        # ...
        self.netG = define_G(opt) # <--- IMPORTANT
        self.netG = self.model_to_device(self.netG)
```

- this one is to call the function to create student instance

```
if self.distillation_type != 'none':
    # ... setup teacher args ...
    self.netTeacher = TeacherModel(**teacher_args).to(self.device)
```

- this is for the teacher instance if the instructions (.json) called for distillation

& For Feature distillation, we also create a "mini model" netP for translation of teacher knowledge to student

Select_network

- Create student and teacher instances

```
def define_G(opt):
    opt_net = opt['netG']
    net_type = opt_net['net_type'] # Reads "swinir_student" from JSON

    if net_type == 'swinir':
        from models.network_swinir import SwinIR as net
        netG = net(...) # Builds Teacher

    elif net_type == 'swinir_student':
        from models.network_swinir_student import SwinIR_Student as net
        netG = net(...) # Builds Student

    return netG
```

- for teacher, it import from
- for student, it import from

network_swinir_student or network_swinir_student_v2 (prof suggestion)

network_swinir

- The pure code of swinir brain infrastructure
- Hierarchy of classes:

(Company) > SwinIR = represent the body, handles images entering or leaving

- `self.conv_first`: The Receptionist. Takes the RGB image (3 channels) and turns it into "Features" (e.g., 60 channels).
- `self.layers`: The Departments. A list of `RSTB` blocks.
- `self.conv_after_body`: The Quality Check. A convolution after the layers are done.
- `self.upsample`: The Expansion Team. Makes the image bigger (x4).
- `self.conv_last`: The Delivery Driver. Turns features back into RGB colors.

The forward function:
It simply pushes the data through these steps in order.
`x = conv_first(x) > x = layers(x) > upsample(x) > conv_last(x)`.

(department) > RSTB = smaller body

1. `self.residual_group`: A team of workers (see Level 3).
2. `self.conv`: A manager that summarizes the team's work.

The "Residual" Magic:
Look at the `forward` function here:

```
Python

return self.conv(self.residual_group(x)) + x
```

It takes the input `x`, processes it, and then **adds the original `x` back.** This is the "Long Skip Connection" specific to this block. It ensures the block learns *corrections* to the image rather than redrawing it from scratch.

(team) > BasicLayer = even smaller body

- It creates a `nn.ModuleList` of `SwinTransformerBlocks`.
- The number of blocks here depends on your config (e.g., `depths=[6,6,6,6]`).
- **Job:** In `forward`, it just loops through them:

```
Python

for blk in self.blocks:
    x = blk(x)
```

(worker) > SwinTransformerBlock = same

1. `self.norm1`: Normalization (cleaning the data).
2. `self.attn`: **Window Attention** (The brain).
3. `self.mlp`: Multi-Layer Perceptron (The muscle/processing).

The Workflow (forward):

1. **Partition:** It chops the image features into small squares (Windows).
2. **Attention:** It runs the windows through `self.attn` (Level 5).
3. **Reverse:** It stitches the windows back together into a full image.
4. **Feed Forward:** It runs the data through `self.mlp` to process the new information.

🚨 **Connection to your Experiment (v2):**
This is exactly where you added your code in `SwinIR_Student_v2`.
You want inside this "Worker" and told it: *"After you finish your job, add the original input again."*

(brain) > Window Attention = pure math

- `self.qkv`: A Linear layer that breaks the input into Query, Key, and Value.
- `self.relative_position_bias_table`: A learnable table that tells the model "Pixel A is to the left of Pixel B."

The Math (Simplified):
It performs the famous equation: $\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d}})V$.

😊 **Newbie Analogy:**
Imagine you are at a crowded party (the Window).

1. **Q (Query):** You yell, "Who here is a Doctor?"
2. **K (Key):** Everyone holding a sign saying their job (Doctor, Baker, Artist).
3. **Softmax:** You ignore the Bakers and focus 99% on the Doctors.
4. **V (Value):** The Doctors tell you medical advice. You absorb that info.

1. The Organization Chart (Who is Who?)

Think of this project as a **Restoration Company**. Here is the staff list:

- **The CEO (`main_train_student.py`):**
  - Runs the show. Reads the business plan (JSON) and tells everyone to get to work.
- **The HR Department (`models/select_model.py` & `select_network.py`):**
  - Hiring managers. They look at the plan and decide which specific workers (networks) and managers (training logic) are needed for the job.
- **The Project Manager (`models/model_plain.py`):**
  - The boss on the floor. He holds the Student and Teacher, conducts the training, compares their work, and calculates the "grade" (Loss).
- **The Genius Consultant (`models/network_swinir.py`):**
  - **The Teacher.** Highly skilled, huge brain (180 width), but expensive and slow. He is there to demonstrate perfect work.
- **The Apprentice (`models/network_swinir_student.py`):**
  - **The Student.** Young, eager, small brain (60 width). He is the one learning.
  - *(Note: `_v2.py` is just a rebellious apprentice who tries a different technique).*
- **The Translator (`netP` inside `model_plain.py`):**
  - A small neural network that simplifies the Genius's complex thoughts so the Apprentice can understand them.
- **Support Staff:**
  - **Librarian (`data/`):** Fetches images from the hard drive.
  - **Photographer (`utils/`):** Saves the progress pictures (`babyx4.png`).

2. The Workflow (A Day in the Life)

Here is exactly what happens, step-by-step, when you run your command.

Phase 1: Setup (The Morning Briefing)

1. **Command:** You run `python main_train_student.py --opt train.json`.
2. **CEO:** Reads `train.json`. Sees `"model": "plain"` and `"net_type": "swinir_student"`.
3. **CEO:** Calls **HR** (`select_model.py`). "Hire a Project Manager."
4. **HR:** Initializes `ModelPlain` (**Project Manager**).
5. **Project Manager:** Wakes up. Reads the JSON.
   - Calls **HR** (`select_network.py`) to build the **Student**.
   - Calls **HR** to build the **Teacher** (and loads his pre-trained brain).
   - If `distillation_type == 'feature'`, hires the **Translator** (`netP`).

Phase 2: The Work Loop (The Grind)

The CEO starts a loop for 20,000 iterations.

1. **Librarian:** Hands over a batch of 32 low-quality images (LR) and 32 high-quality images (HR).
2. **CEO:** Passes them to the **Project Manager**.
3. **Project Manager:**
   - **Step A:** Gives LR images to **Student**. "Fix this."
     - *Student:* Processes image layers 1, 2, 3, 4. Saves intermediate features (thoughts). Outputs Final Image.
   - **Step B:** Gives LR images to **Teacher**. "Show us how it's done."
     - *Teacher:* Processes image layers 1...6. Saves intermediate thoughts. Outputs Final Image.
4. **Project Manager (Grading):**
   - **Pixel Grade:** Compares Student Image vs. HR Image.
   - **Distillation Grade:** Compares Student Image vs. Teacher Image.
   - **Feature Grade (Model C):**
     - Takes Teacher's thoughts.
     - Passes them through **Translator**.
     - Compares Translated Thoughts vs. Student's Thoughts.
5. **Correction:** The Project Manager yells at the Student (Backpropagation), adjusting the Student's and Translator's neurons to reduce errors. **The Teacher is never scolded (frozen).**

Phase 3: Review (The End of Day)

Every 2,000 steps:

1. **Photographer:** Takes the Student's current best attempt on a test image (`baby.png`) and saves it.
2. **Project Manager:** Saves the Student's current brain state into a `.pth` file.