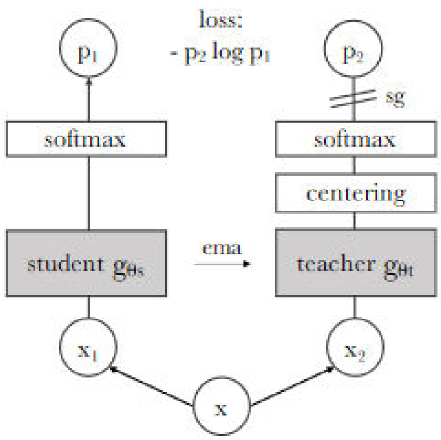
DINO - a method for unsupervised pre-training of ViTs

- **Self-Distillation with no labels**
- The transformer doesn't get taught what a class is, nor is it taught any form of image segmentation
- Can track objects behind occlusion
- If you use DINO for image classification and you view the **feature representation**, it will **cluster images of the same class** together, and it will also **cluster clusters of similar classes together**
  - **Feature Space -**
- *Useful for things like image retrieval (similar images clustered together), classification, zero-shot classifier (by doing knn classifier), etc*
- It does all of this with no supervision



Architecture

- They have a **student+teacher**
  - This provides the **self-distillation part**
  - **Momentum teacher -**
- The output of the teacher goes through **centering**, followed by a **softmax**
  - Keeps the model from **mode collapse**
- There's **no** use of **batch-norm**

Self-Supervised Learning - you have **no labels**

- In this paper, you don't have *negative sample mechanism* or *contrastive learning mechanism*
  - **Negative Sampling Mechanism -**
  - **Contrastive Learning Mechanism -**
    - You take a few patches, A, a (say 2)
    - You then take a patch from another image b
    - Patch **A** is your **anchor**, and the model has to decide which patch **a** or **b** is in the image with **A**
    - The model learns what kind of stuff is likely to be in the same image
- **We want** a model that gives us **sensible representations**
  - Difficult with no labels
- We **augment images** in different ways
  - You do random perturbations to an image
  - You might flip it, add jitter, add some **solarisation** (dark parts lighter, light parts darker)

Teachers only have **global cropping**
Students can have **both**

- This means that whenever the student gets a small crop and teacher a large crop, the student has to learn that whatever small patch it sees is something part of a larger thing a teacher sees
  - Different to **contrastive learning**, you look at it **deeper** than just using **negative samples**
    - You have to say what a small patch **describes as a whole and differentiates it from others**

Paper

- Utilises different augmentations
- You can also add **crops**
  - **Global Crops -** cover more than 50% of the image
  - **Local Crops -** less than 50% of the image

Student-Teacher (Self-Distillation)

- We take an image and we **make two different augmentations of the same image**
- Make **two** different **versions** of it
- *We want the two models to output the same output for different augmentations of the same image*
- Our goal is to **minimise loss = $-p_2 \log p_1$**
- We want the student and the teacher to **predict the same class output**
- *Theoretically, the easiest method would be to put them through the same transformer network*
  - Can't pass the images through the same network because they would start predicting the same class
    - This is called a **collapse -** when the model models everything to the **same representation**
- This idea comes from **distillation -** You have a **big model (teacher)** and you want to make the model **smaller**
  - You transfer the knowledge from the teacher model to the student model
  - Works better than teaching the student from scratch
- **Self-Distillation -** The two networks are of the same architecture
  - We **only train the student**
  - The **teacher** is constructed **from the student**
  - We train the student on the teacher until they predict the same thing, then **construct the teacher from the student** using **ema**
- **Exponentially moving average EMA -** we keep the teacher model, and as we update the student, **we move the teacher a little bit in the direction of the student model**
  - There is a scale associated with the step size
  - **Lots of hyperparameters**

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

- **T1 =** output of **x1** going through the **teacher**
- **S2 =** output of **x2** going through the **student**

Centering and Softmax

- You force the model to come up with a **k-dimensional** classification problem by itself
- *It chooses what the classes are so it has to make representations that **allows itself to come up with a problem that it can solve**, and centering + softmax makes sure it goes well and doesn't go into collapse*

Centering **- The teacher** keeps **a running average of all of the representations** that the teacher sees

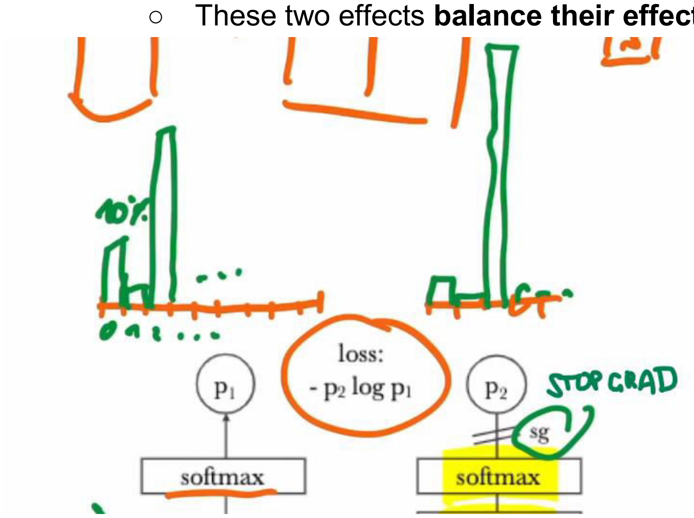
- You **subtract** the average from the **logits**
    - Avoids collapse
    - Kind of like a **normalisation**
    - It keeps the **logits** be in a range that is manageable and have some variance
        - Since the student learns from the teacher, this also affects the student
- **Centering** prevents one dimension from dominating (always predicting the same class), however, this causes **collapse to a uniform distribution**
- Centering is only applied batch-wise and is like adding a **bias term** to the teacher

Softmax (Sharpening) -

- **Temperature Parameter -** the softmaxes in both networks have a temperature parameter
    - The two temperature parameters **are different**

    $$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K}\exp(g_{\theta_s}(x)^{(k)}/\tau_s)},$$

    -
        - The **temperature parameter T_s** for the **teacher** is much **lower**
            - This is called **sharpening**
    -
    -
- *Interesting they have softmax, couldn't they have just used l2-distance between logit representations*
- Here we use **cross entropy**
    - Softmax outputs a **normalised distribution**
    - You choose any number **k** to be the size of output
    - *The softmax will be a **distribution over the size of output** (where each node is a class)*
    - The teacher has **sharpening**, meaning it will have a more **peaked distribution**
        - Student gets a **less noisy, stronger signal**
- **Sharpening** has the opposite effect of **centering**
    - These two effects **balance their effects**



Conclusion

- We **augment** the inputs in **two random ways**
- We put all of the images **through the student and the teacher**
- The teacher is an **ema of the student**
- This gives us **different representations of different augmentations** of the same image
- We put the **representations through a softmax** and force the output **distribution to be the same**
- The **teacher centers** the logits by an **exponential running average** of all of the representations it has seen
- The teacher also has a **sharper softmax**
- This results in **good representations and avoid collapse**

Results

- We can now do **K-Nearest Neighbour classification**
    - A type of **zero-shot learning**
- **Image Retrieval**
- **Linear classification** on top of the representations
- **Copy Detection -** you wanna realise if someone made another image out of an original image
- **Ablation -** When comparing the attention heads for a ViT trained in a supervised way vs Dino, the Dino attention heads are way cleaner

The class representations come from the **CLS** token

- *You can visualise the attention heads of the* ***CLS token****, you get really strong semantic segmentation maps*
- A possible reason why **Dino attention is cleaner than supervised** is that:
    - Supervised models tend to stop learning when they master a problem
    - There is a problem of **shortcut learning**
    - The extra noise in attention maps is probably caused by **extra optimisations** that it tries to make that works overall for the whole dataset
    - There is no **hyper-optimisation** on a single task

Why does this work?

- **Augmentations** are very important (i.e. the **multicrop augmentation** was very important)
    - **Multicrop Augmentation -**
    - The **augmentations** are essentially where you put the **human prior**
        - That's where you tell the model what to pay attention to and what to ignore
    - To do fully unsupervised, **domain agnostic** learning, we should replace **human designed augmentations**
    - Relies on **cropping** which is teaching the model to represent images **invariant to cropping position**
- **Dataset -** Datasets consist of **object-centered images** that always have an **object of interest**
    - **Strong Inductive Prior -** produces representations that focus on the objects of interests
    - *Makes sure that the* ***cropped images don't have*** *different objects that have* ***little in common***
    - As images are taken by humans, and you have to scrape the dataset somehow, the way you are creating a dataset is causing an **implicit bias** that affects where the **attention goes**
        - DINO probably relies a lot on **dataset construction**
        - We shouldn't expect this model to work on random pictures of the world, because the photos of the dataset used to train it weren't **IID**