期末大作业讲评

MovieLens上的跨模态对齐任务

王瑞环

希望能通过这次作业...

- 回顾一些课程传达的东西
  - 数据科学之“道”

    普适性的流程—数据观察、预处理、建模、模型优化、测试与分析、可视化...
  - 问题建模之“法”

    个性化的理解—数据处理的设计、建模设计、损失函数设计、训练模式的设计...
  - 编程实现之“术”

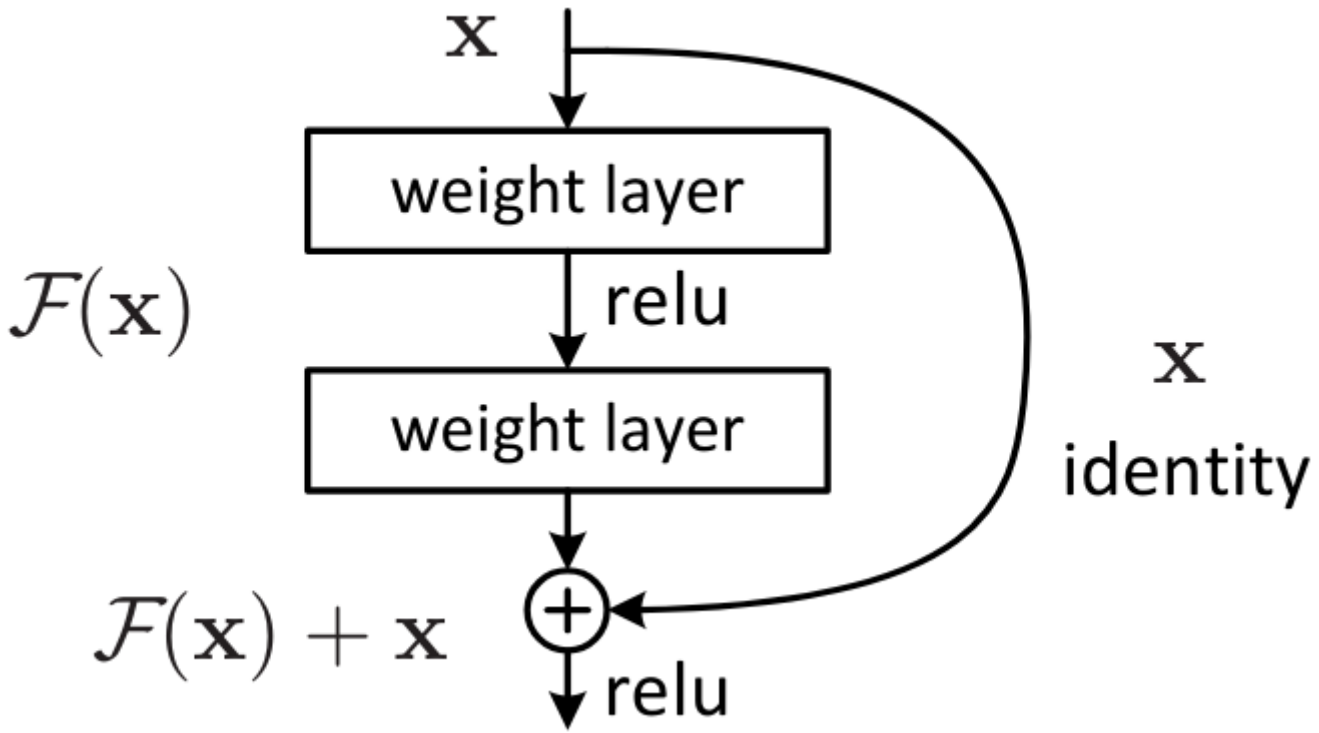
    强有力的工具—pytorch神经网络、预训练模型...
- 尝试一些更为新鲜的东西
  - 分类、回归、聚类并不是世上仅有的全部任务
  - 也许要比Loss(y_pred, y_truth)更灵活的损失函数设计
  - 比刷指标更有意思的是用训好的模型更细致地“玩一玩”数据
  - 深度学习新版本—预训练模型体验服

Part1. “建设武装力量”

——能打蚊子的除了巴掌，还有大炮

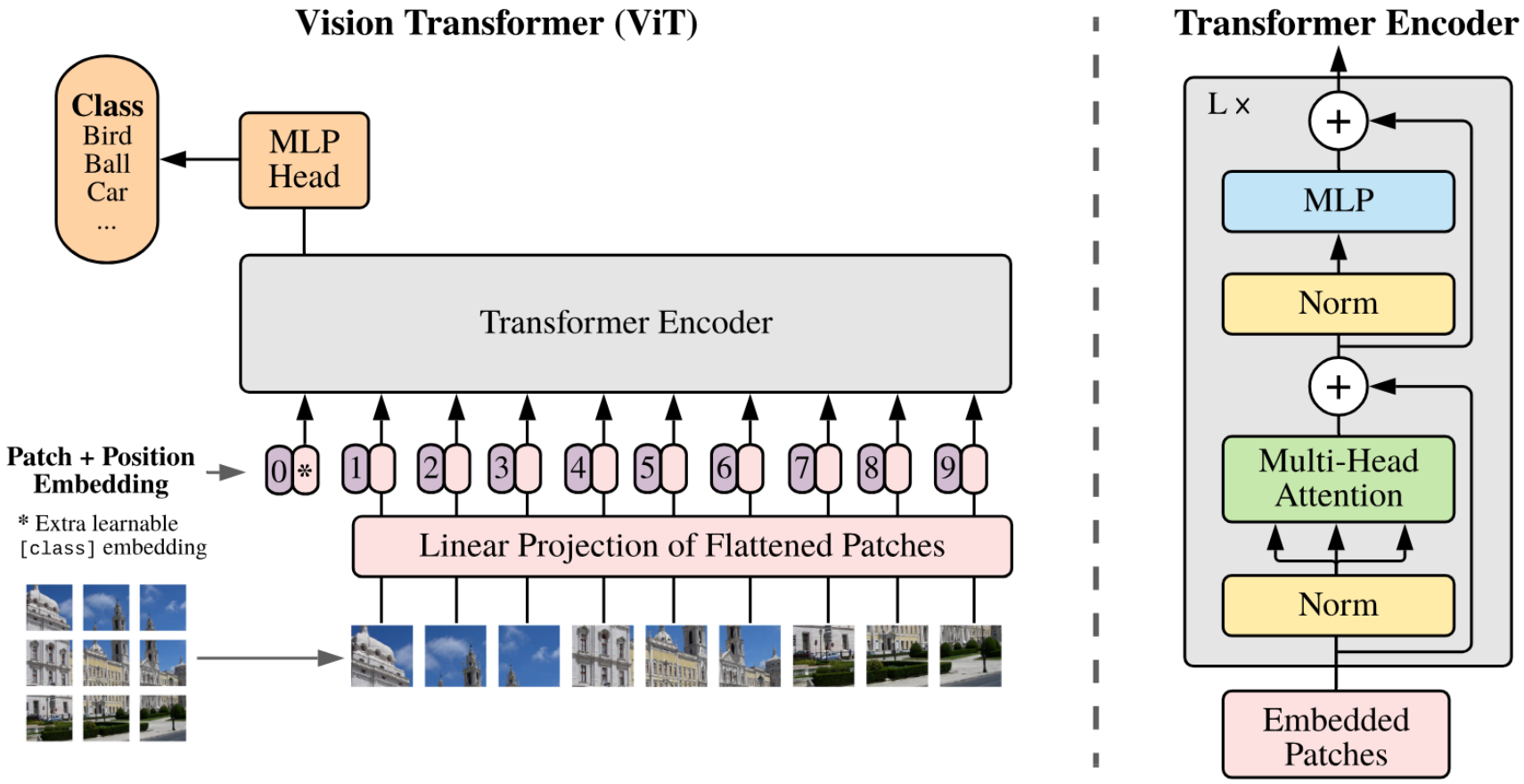
ResNet

- Paper, torchvision, Hugging Face



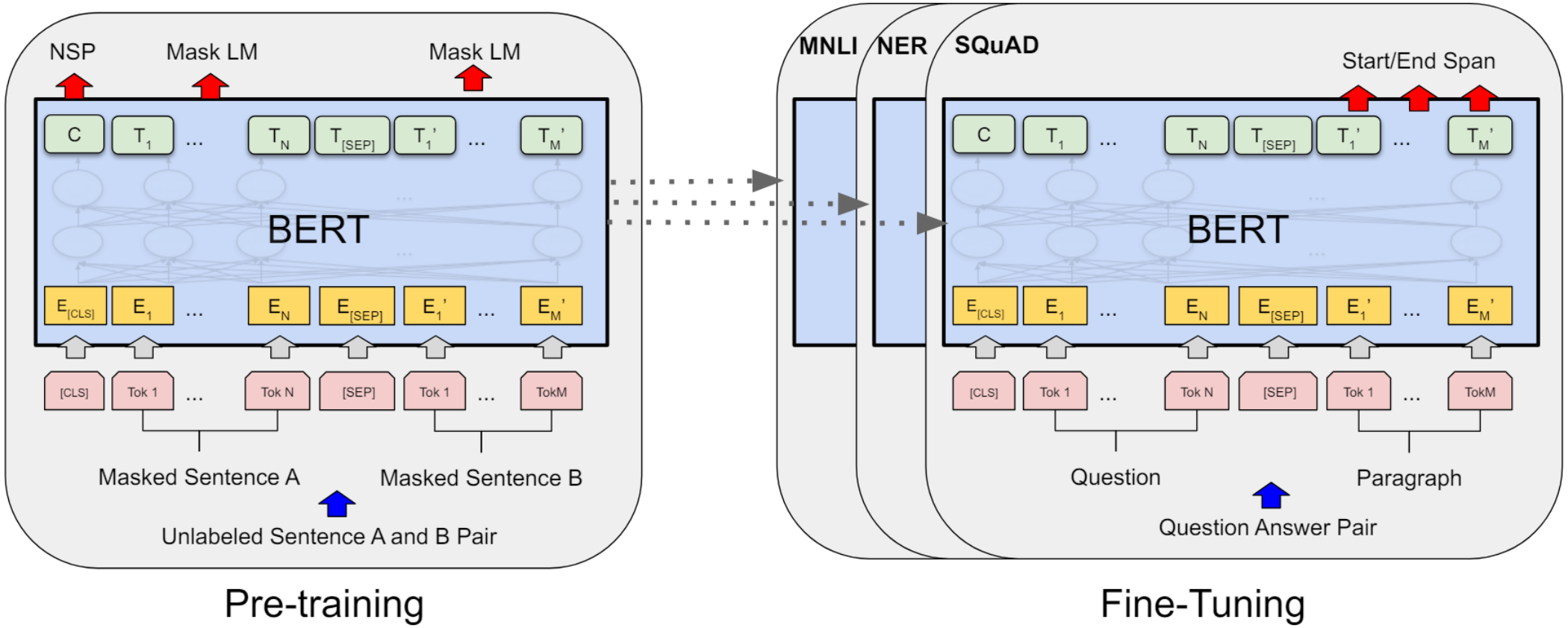
ViT

- Paper, torchvision, Hugging Face



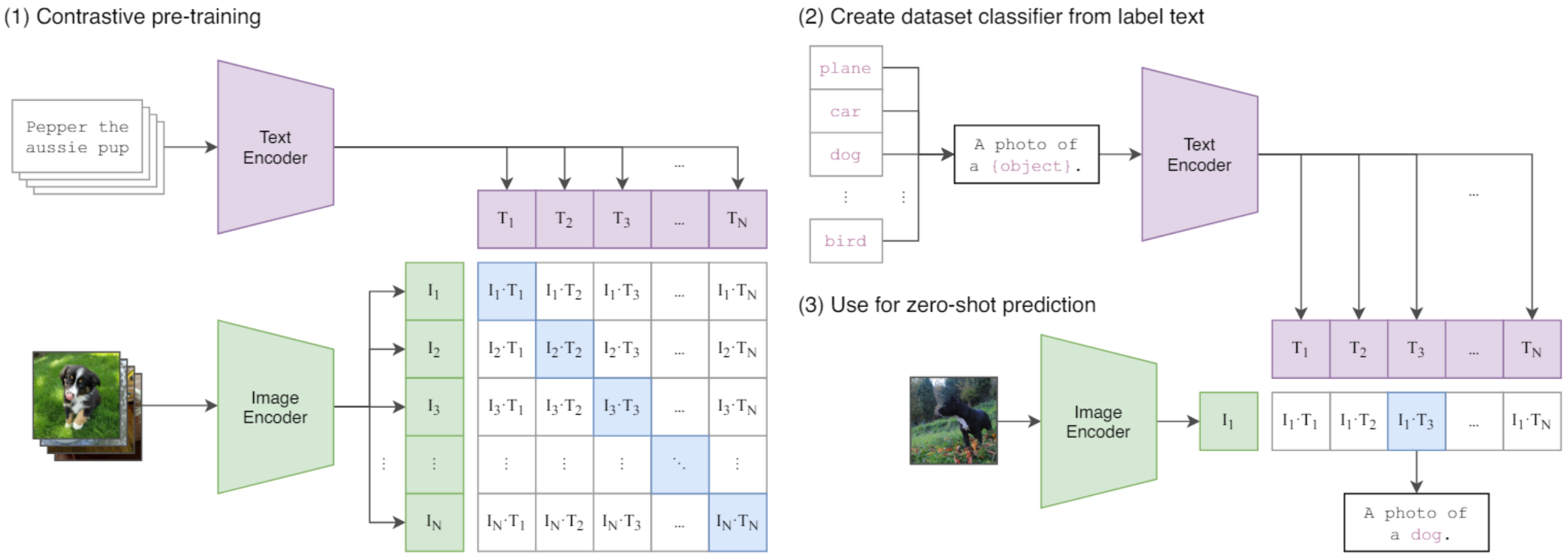
BERT

- Paper, GitHub, Hugging Face



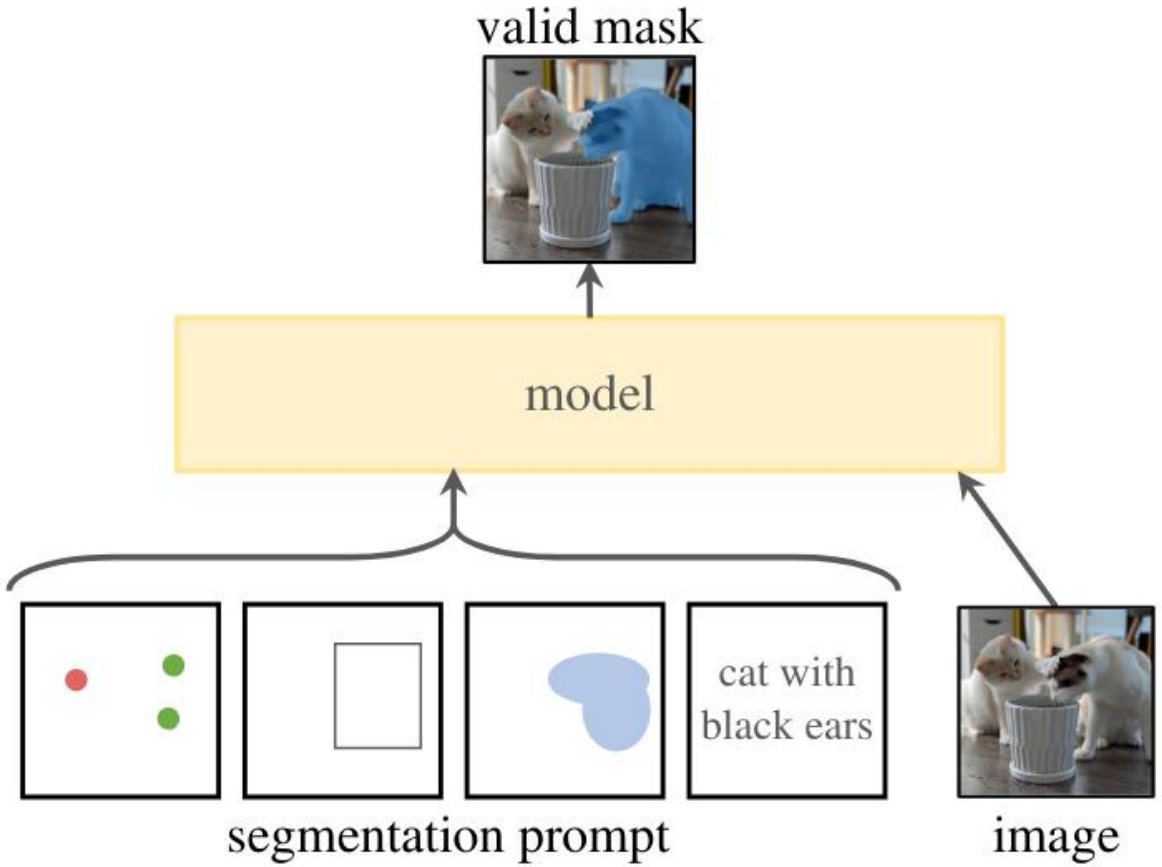
CLIP

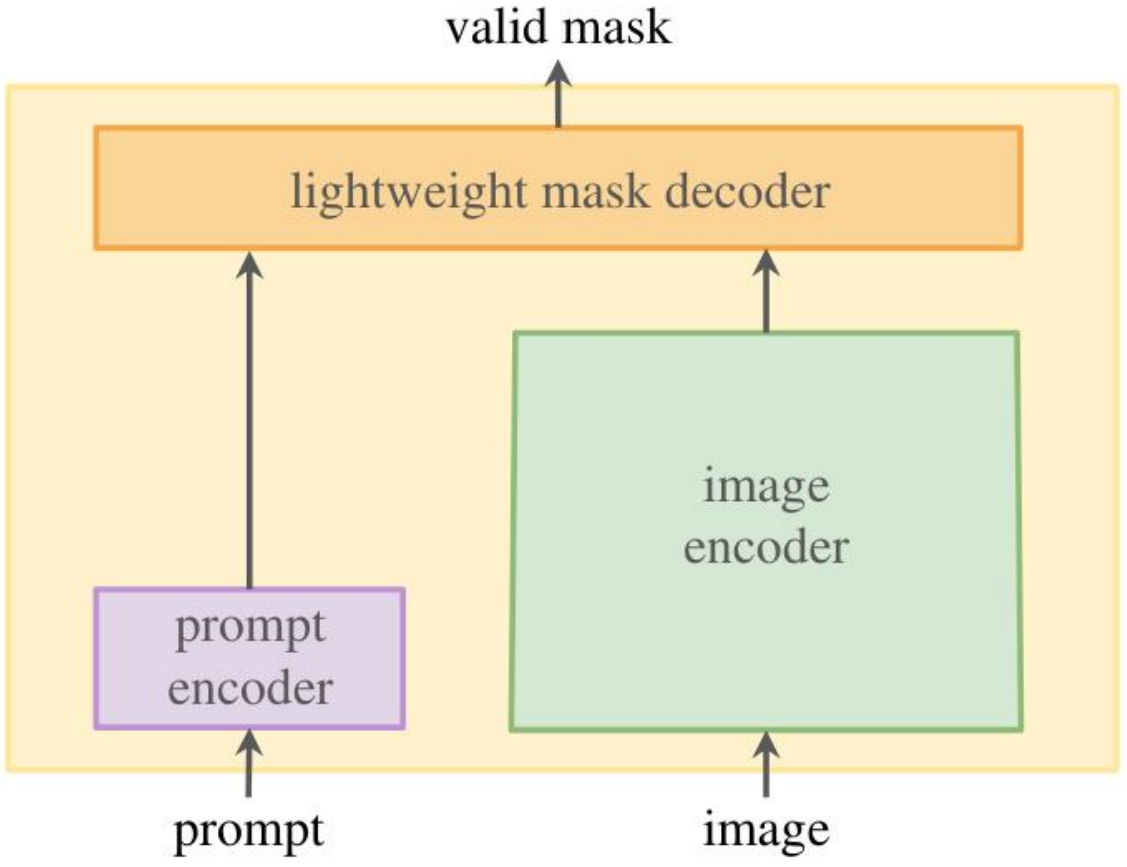
- HomePage, Paper, GitHub, Hugging Face



SAM

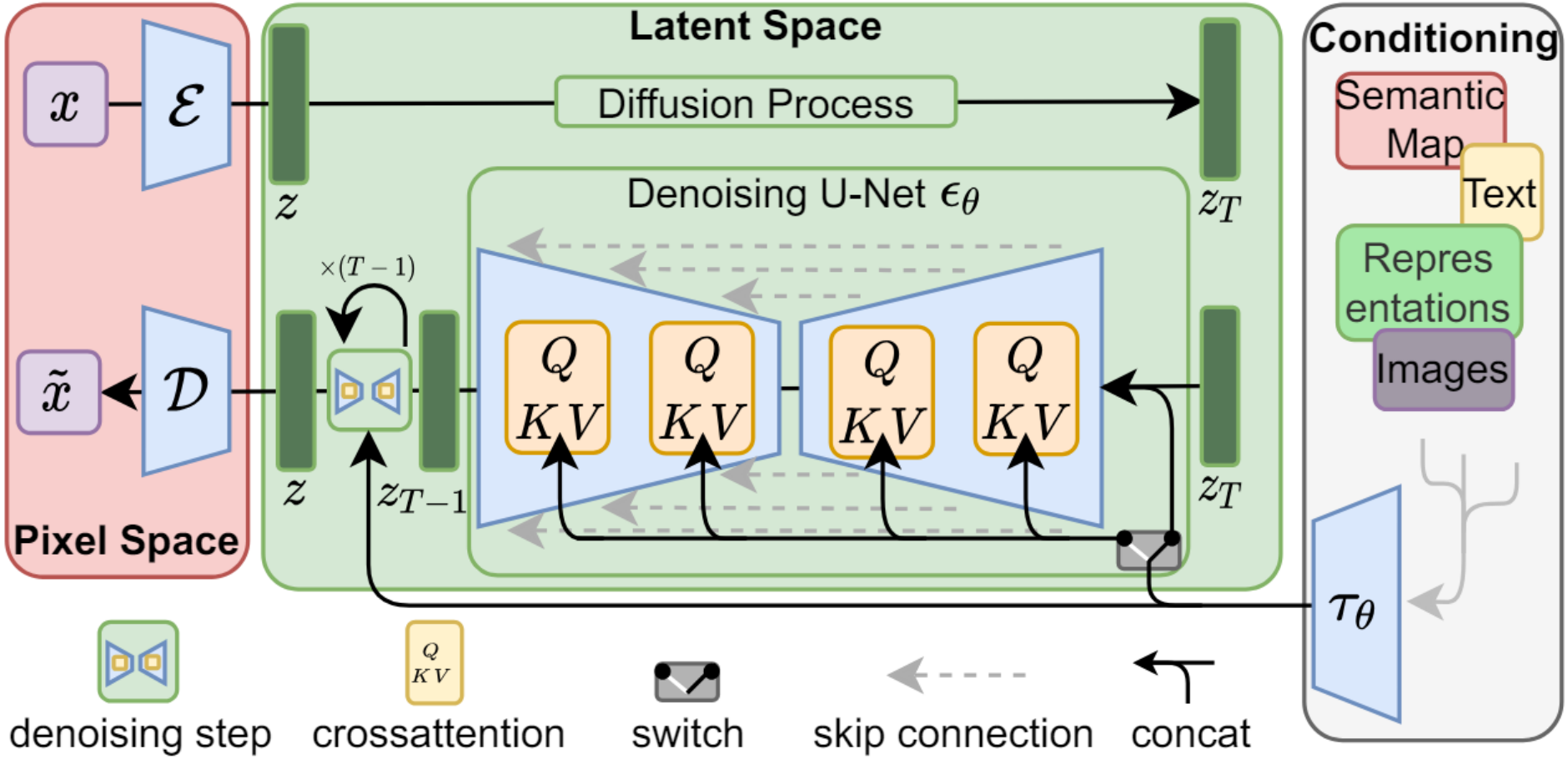
- [HomePage](), [Paper](), [GitHub](), [Hugging Face]()



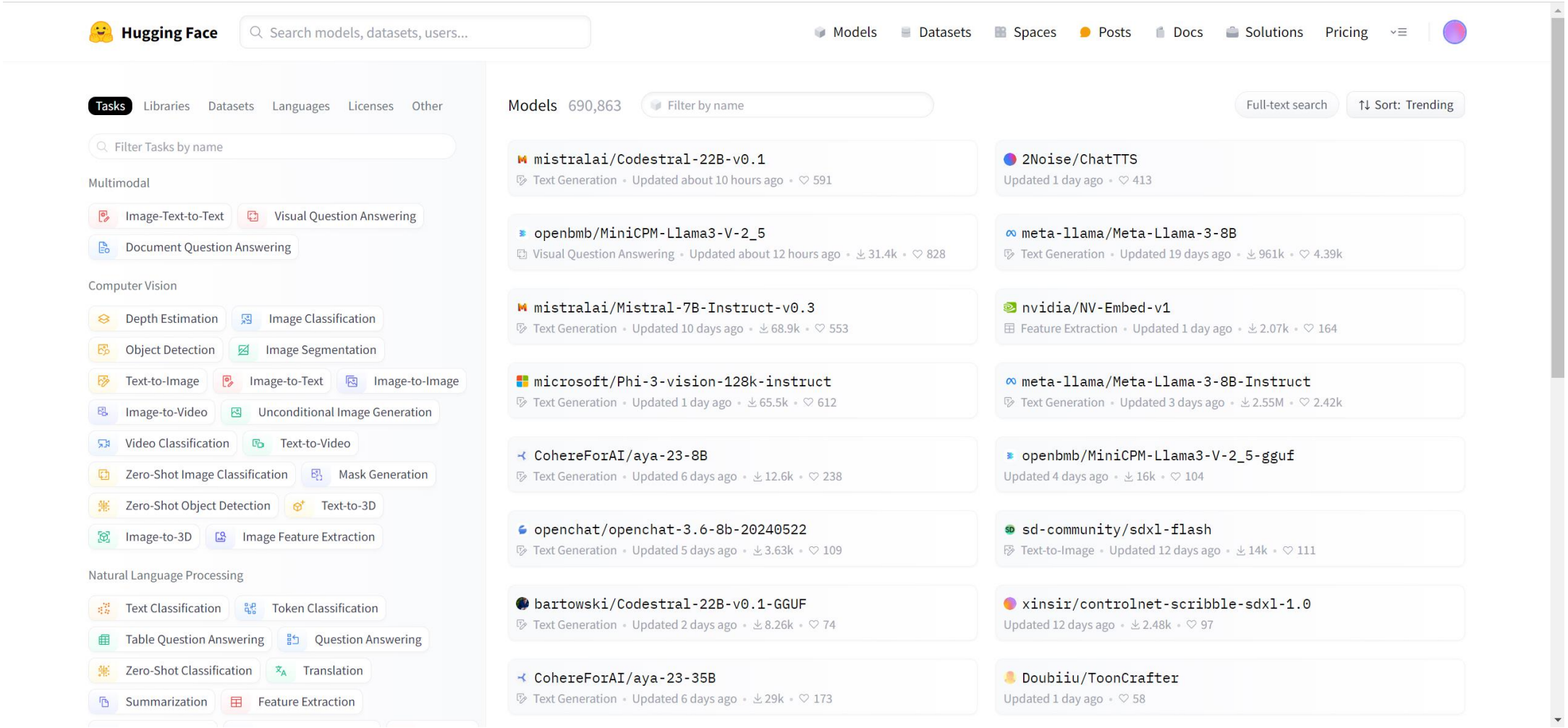


Diffusion Models

- HomePage, Paper, GitHub, Hugging Face



Hugging Face — 超级武器库



PapersWithCode — 研发成果展

Search

Browse State-of-the-Art | Datasets | Methods | More | Sign In

Top | New | Greatest

Trending Research

Subscribe

MiniCPM-V & OmniLMM

LLaVA-UHD: an LMM Perceiving Any Aspect Ratio and High-Resolution Images

openbmb/minicpm-v • PyTorch • 18 Mar 2024

To address the challenges, we present LLaVA-UHD, a large multimodal model that can efficiently perceive images in any aspect ratio and high resolution.

★ 4,440

6.68 stars / hour

Paper

Code

LightAutoML: AutoML Solution for a Large Financial Services Ecosystem

sb-ai-lab/lightautoml • PyTorch • 3 Sep 2021

We present an AutoML system called LightAutoML developed for a large European financial services company and its ecosystem satisfying the set of idiosyncratic requirements that this ecosystem has for AutoML solutions.

AutoML

★ 1,051

1.52 stars / hour

Paper

Code

kaggle — 试验田

kaggle

Search competitions

Filters

Create

Home

Competitions

Datasets

Models

Code

Discussions

Learn

More

View Active Events

All competitions | Featured | Getting Started | Research | Community | Playground | Simulations | Analytics

Active Competitions

Hotness

AI Mathematical Olympiad - Progress Prize 1

Solve national-level math challenges usin...

Featured · Code Competition

950 Teams

$1,048,576 a month to go

LMSYS - Chatbot Arena Human Preference...

Predicting Human Preferences in the Wild

Research · Code Competition

552 Teams

$100,000 2 months to go

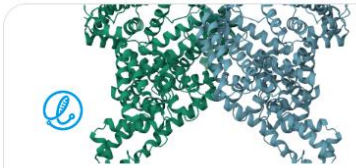
Learning Agency Lab - Automated Essay Scoring...

Improve upon essay scoring algorithms to...

Featured · Code Competition

1883 Teams

$50,000 a month to go



Leash Bio - Predict New Medicines with BELKA

Predict small molecule-protein interaction...

Featured

1206 Teams

$50,000 a month to go

Image Matching Challenge 2024 - Hexathlon

Reconstruct 3D scenes from 2D images o...

Research · Code Competition

986 Teams

BirdCLEF 2024

Bird species identification from audio, foc...

Research · Code Competition

850 Teams

LEAP - Atmospheric Physics using AI (ClimSim)

Simulate higher resolution atmospheric pr...

Research

493 Teams

USPTO - Explainable AI for Patent Professionals

Help patent professionals understand AI r...

Featured · Code Competition

162 Teams

在代码中使用预训练模型

- Ref. 赵凌哲，使用torchvision中的ResNet

```
import torchvision.transforms as transforms
from torchvision.models import resnet50

resnet_model = resnet50(pretrained=True)
resnet_model = torch.nn.Sequential(*(list(resnet_model.children())[:-1]))
resnet_model.eval()
#ResNet-50

img_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
#预处理，为了满足ResNet的要求

I_f = []
for path in all_posters_filename:
    poster = Image.open(os.path.join('./poster', path))
    if poster.mode != 'RGB':
        poster = poster.convert('RGB')
    poster_processed = img_transform(poster).unsqueeze(0)
    with torch.no_grad():
        features = resnet_model(poster_processed)
    I_f.append(features.squeeze().detach().numpy())
```

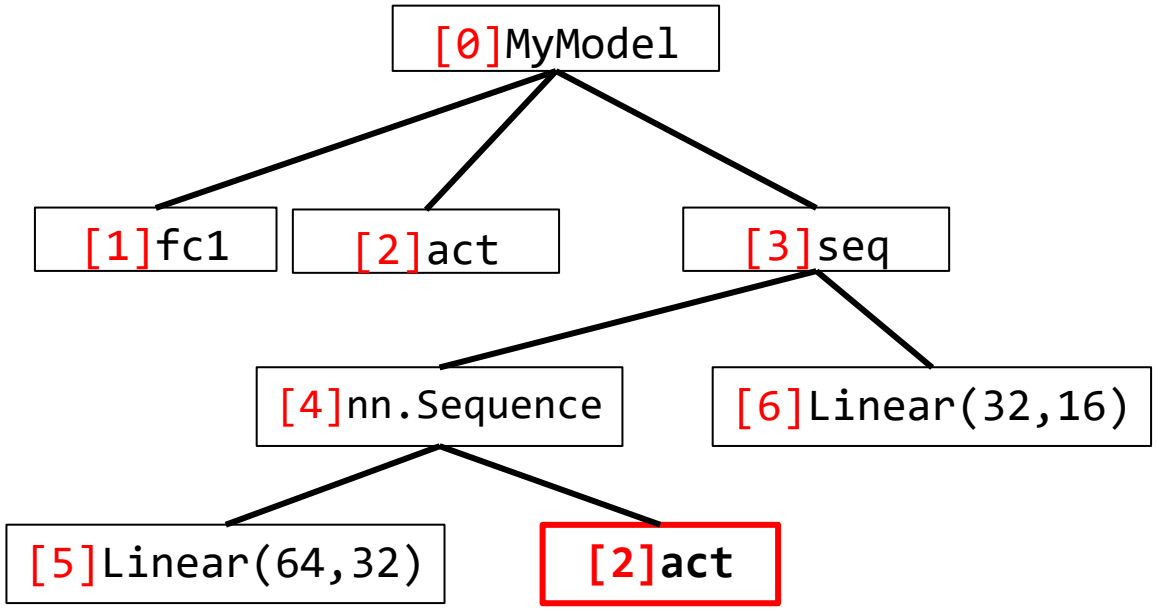
- nn.Module.children()
  - 后一页详细讲
- train()和eval()模式
  - BatchNorm层的表现不同
  - Dropout层的表现不同
- with torch.no_grad()
  - 在小数据上微调预训练模型，通常固定大部分参数，只修改最后一层
  - 不计算梯度可以加快速度

nn.Module.children()

nn.Module.modules()

- 深度优先遍历

```python
class MyModel(nn.Module):
    def __init__(self, in_channels=128, out_channels=16):
        super().__init__()
        self.fc1 = nn.Linear(in_channels, 64)
        self.act = nn.ReLU()
        self.seq = nn.Sequential(
            nn.Sequential(nn.Linear(64, 32), self.act),
            nn.Linear(32, out_channels)
        )
```




```python
for i, c in enumerate(mymodel.modules()):
    print(f'[{i}]', c)
```

```
[0] MyModel(
  (fc1): Linear(in_features=128, out_features=64, bias=True)
  (act): ReLU()
  (seq): Sequential(
    (0): Sequential(
      (0): Linear(in_features=64, out_features=32, bias=True)
      (1): ReLU()
    )
    (1): Linear(in_features=32, out_features=16, bias=True)
  )
)
[1] Linear(in_features=128, out_features=64, bias=True)
[2] ReLU()
[3] Sequential(
  (0): Sequential(
    (0): Linear(in_features=64, out_features=32, bias=True)
    (1): ReLU()
  )
  (1): Linear(in_features=32, out_features=16, bias=True)
)
[4] Sequential(
  (0): Linear(in_features=64, out_features=32, bias=True)
  (1): ReLU()
)
[5] Linear(in_features=64, out_features=32, bias=True)
[6] Linear(in_features=32, out_features=16, bias=True)
```

在代码中使用预训练模型

- Ref. 李之怡，使用timm上的ResNet

```
import timm
from torchvision import transforms
from PIL import Image
transform = transforms.Compose([
        transforms.Resize((224, 224)), transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])
def preprocess_poster(poster_path: str) -> torch.Tensor:
    image = cv2.imread(poster_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = Image.fromarray(image)
    image = transform(image)
    return image
class ImageEncoder(nn.Module):
    def __init__(self):
        super(ImageEncoder, self).__init__()
        self.model = timm.create_model("resnet50", pretrained=True, num_classes=0, global_pool="avg")
        for param in self.model.parameters():
            param.requires_grad = False
    def forward(self, x):
        return self.model(x)
```

在代码中使用预训练模型

- Ref. 沈千帆，使用Hugging Face上的ViT

```
from transformers import ViTModel, ViTFeatureExtractor
cache_dir = './hub'
class MovieLensDataset(Dataset):
    def __init__(self, data_root, posters, intros, **kwargs):
        ...
        self.image_processor = ViTFeatureExtractor.from_pretrained(
            'google/vit-base-patch16-224', cache_dir=cache_dir)
        ...

    def process_poster(self, path):
        poster = cv2.imread(path)
        poster = cv2.cvtColor(poster, cv2.COLOR_BGR2RGB)
        poster = self.image_processor(images=poster, return_tensors='pt')['pixel_values'].squeeze(0)
        return poster

     ...

class MyModel(nn.Module):
    def __init__(self, hidden_dim):
        super(MyModel, self).__init__()
        self.image_encoder = ViTModel.from_pretrained('google/vit-base-patch16-224', cache_dir=cache_dir)
```

在代码中使用预训练模型

- Ref. 杨明，使用Hugging Face上的BERT

```
from transformers import BertTokenizer, BertModel
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
# 加载预训练的BERT模型和分词器
Bert_path = "..."
tokenizer = BertTokenizer.from_pretrained(Bert_path)
model = BertModel.from_pretrained(Bert_path).to(device)

# 将简介转换为BERT输入格式
def get_bert_embeddings(intro):
    inputs = tokenizer(intro, return_tensors='pt', truncation=True, padding=True, max_length=512).to(device)
    with torch.no_grad():
        outputs = model(**inputs)
    # 提取[CLS]标记的特征向量
    cls_embeddings = outputs.pooler_output.cpu()
    return cls_embeddings.squeeze().numpy()

# 获取所有简介的特征
intro_features = df_movie_info['intro'].apply(get_bert_embeddings)
```

在代码中使用预训练模型

- Ref. 肖旭森，使用Hugging Face上的CLIP

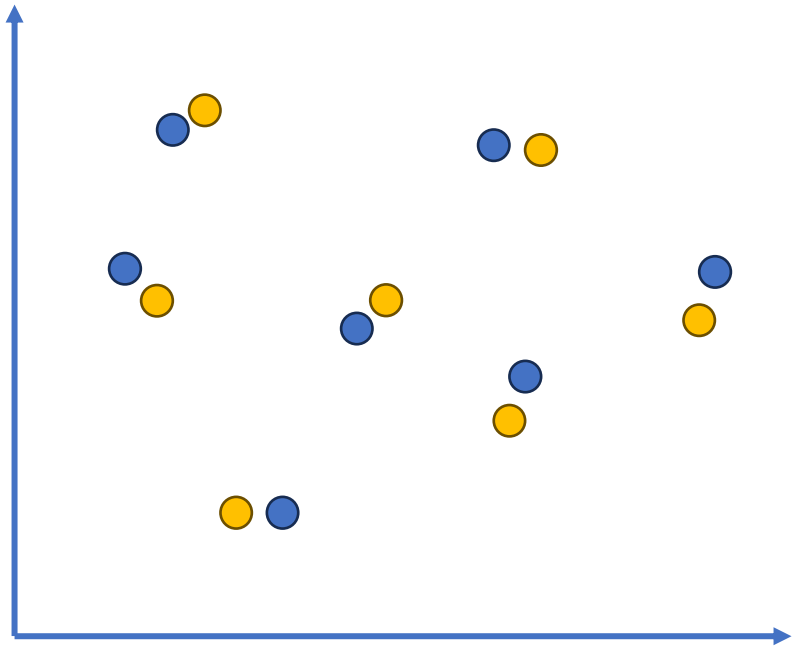
```
from sentence_transformers import SentenceTransformer, util
from PIL import Image

def get_dist(posters_path_list, data_intros, model='clip-ViT-B-32'):
    model = SentenceTransformer(model)

    images = [Image.open('./poster/'+poster_path) for poster_path in posters_path_list]
    image_embeddings = model.encode(images, convert_to_tensor=True, show_progress_bar=True)

    intro_embeddings = model.encode(data_intros, convert_to_tensor=True, show_progress_bar=True)
    cos_sim = util.cos_sim(image_embeddings, intro_embeddings)

    distances = 1 - cos_sim
    return distances

D_test = get_dist(posters_test, intros_test)

Accuracy on dim 0: 65.08%
Accuracy on dim 1: 54.19%
```

Part2. 瞄准目标

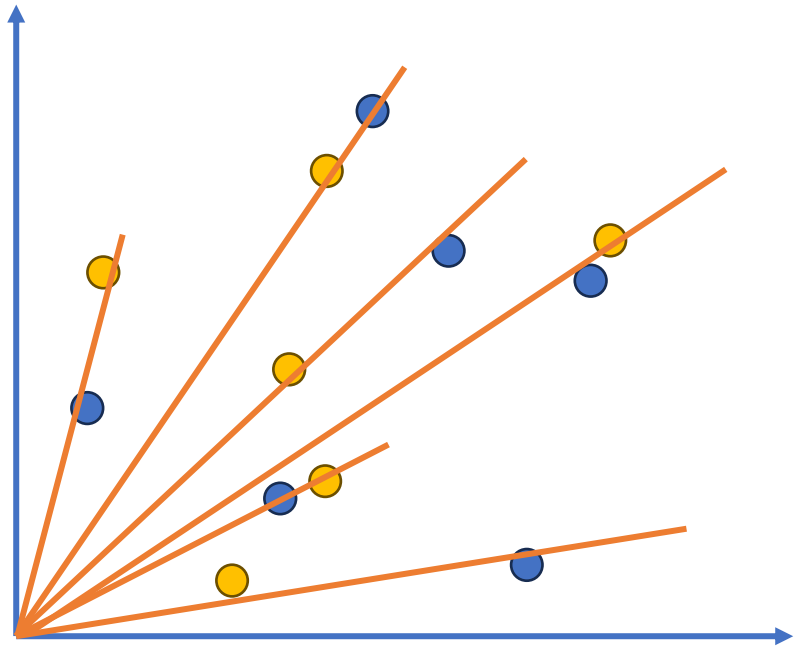
——方向打反了核弹也没用

目标是什么

- 编码后的特征空间应该长什么样?



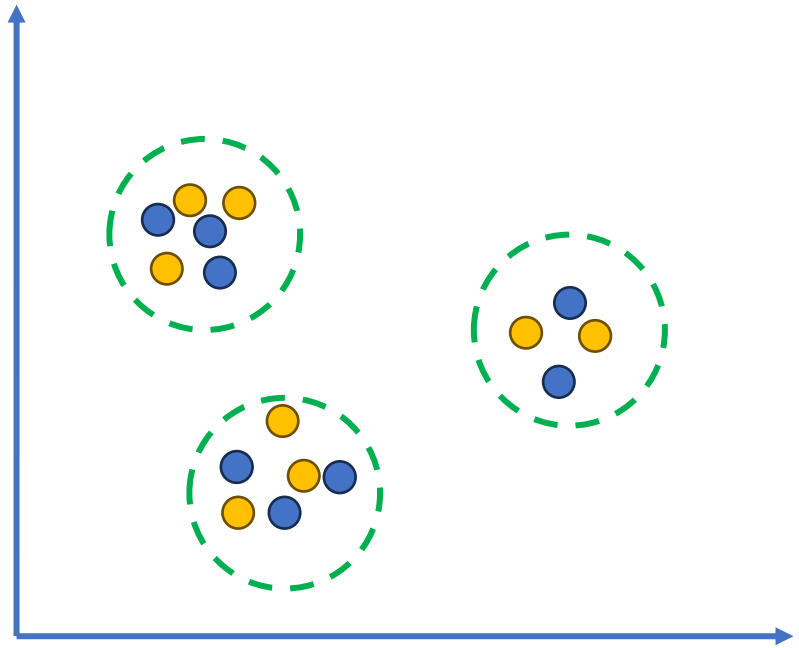
同一个电影的海报和简介特征距离相近



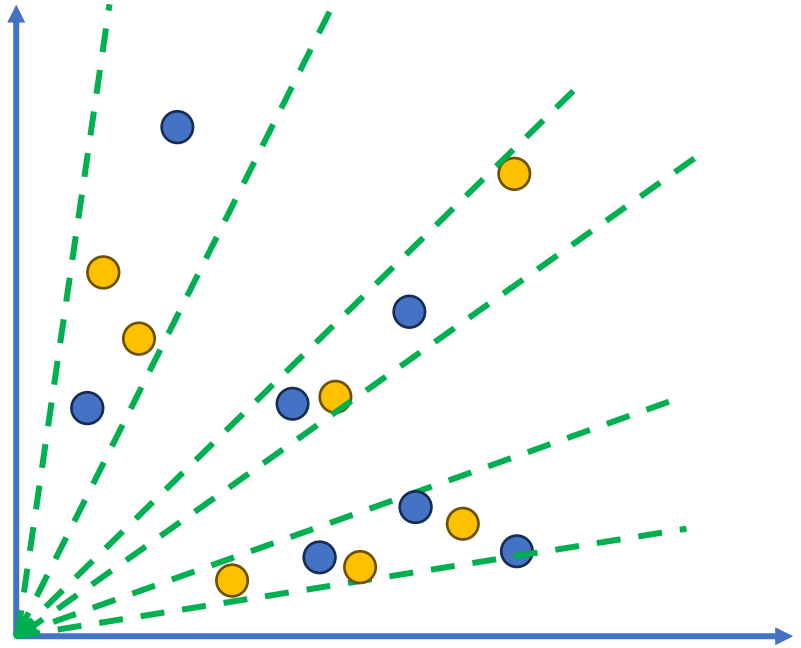
同一个电影的海报和简介特征夹角相近

目标是什么

- 也许会有更丰富的结构



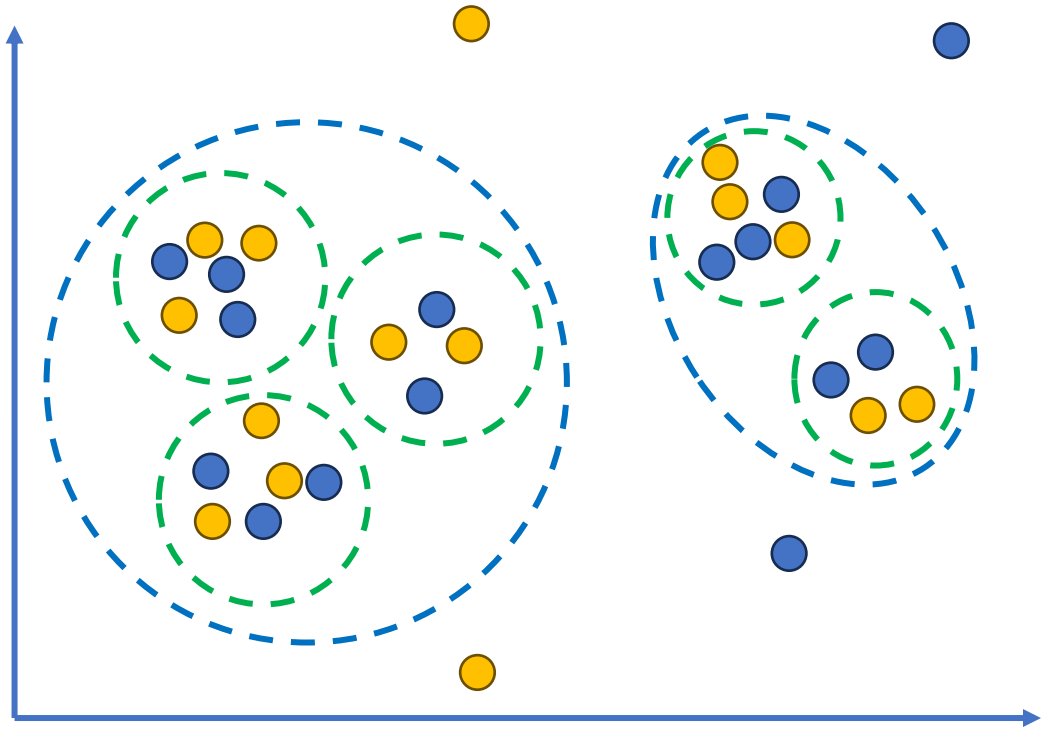
同一**种**电影的海报和简介特征距离相近



同一**种**电影的海报和简介特征夹角相近

目标是什么

- 结构的粒度



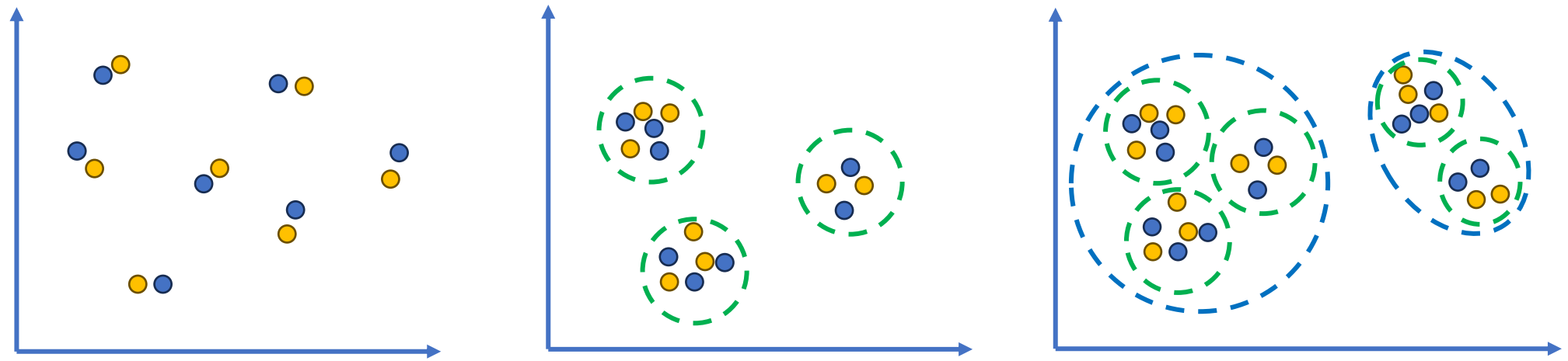
同一**大类**电影的海报和简介特征距离相近

在某一大类中，同一**小类**电影的海报和简介特征距离**更**近

如果数据足够丰富，也许能分出更多等级

当然，肯定也少不了噪声和异常数据

目标是什么



拉近属于同一**个**电影的特征
推远属于不同电影的特征

拉近属于同一**种**电影的特征
推远属于不同**种**电影的特征

- 种类信息从何而来?
  - 无监督模式：通过聚类赋予标签
  - (半)监督模式：训练数据中加入类别标签

前往新目标——从回归出发

- 面对一个新任务，如何设计模型和损失函数以达成任务目标?
- 从已经接触过的任务中找灵感!
- 回归任务的策略

$$\underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{X,y}[(f_\theta(X) - y)^2]$$

- 当然，也可以不用欧式距离

$$\underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{X,y}[d(f_\theta(X), y)\,]$$

- 改编成跨模态对齐任务的策略

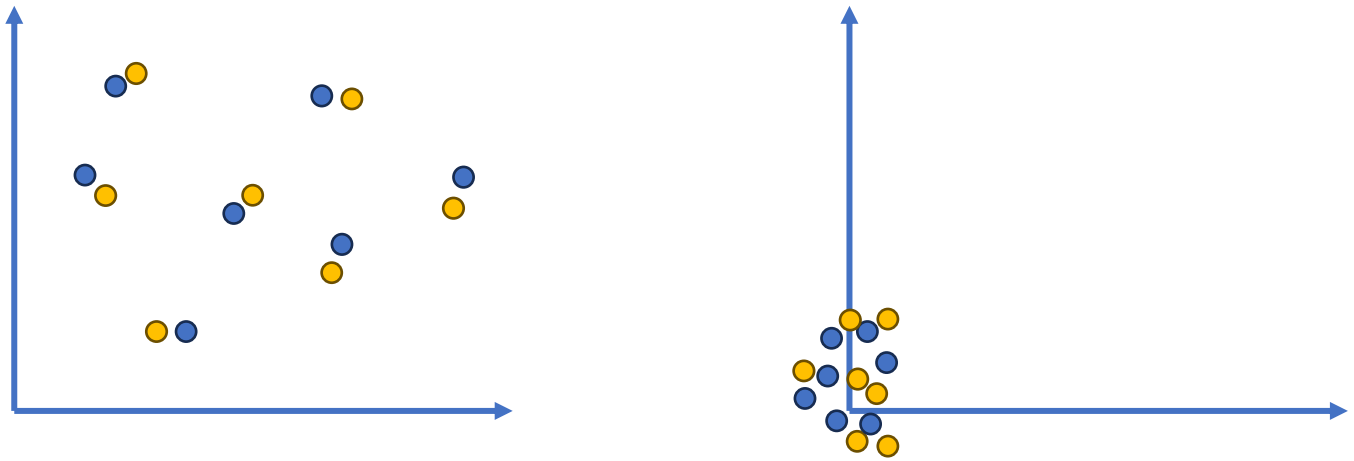
$$\underset{\theta_1,\theta_2}{\operatorname{argmin}}\, \mathbb{E}_{X,Y}\left[d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right)\right]$$

前往新目标——从回归出发

- 这个策略的不足之处

$$\underset{\theta_1,\theta_2}{\operatorname{argmin}}\ \mathbb{E}_{X,Y}\left[d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right)\right]$$

- 考虑d为欧氏距离，若$f(X) \equiv 0, g(Y) \equiv 0$，则损失函数同样可以达到最小值0，但是这显然不是我们想要的结果
- 拉近属于同一电影的特征，**推远**属于不同电影的特征



前往新目标——从回归出发

- 如何推远？最直接的想法，当X, Y'属于不同电影时

$$\underset{\theta_1,\theta_2}{\operatorname{argmax}}\, \mathbb{E}_{X,Y'}\left[d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right)\right]$$

- 新的问题：不加约束的max是否太激进了？直观来看
  - 如果d采用欧氏距离，那么属于不同电影的特征会被推往**无限远**
  - 如果d采用余弦距离（$d(x,y) = 1 - \cos < x, y >$），那么属于不同电影的特征会被推往**完全相反**的方向
- 如何解决：从“尽量远”到“足够远”，只要距离达到某个阈值α，就认为已经被完全分离

$$\underset{\theta_1,\theta_2}{\operatorname{argmax}}\, \mathbb{E}_{X,Y'}\left[\min\left(\alpha, d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right)\right)\right]$$

前往新目标——从回归出发

- 进一步推导

$$\underset{\theta_1,\theta_2}{\operatorname{argmax}}\, \mathbb{E}_{X,Y'}\left[\min\left(\alpha, d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right)\right)\right]$$

$$\underset{x}{\operatorname{argmax}}\, f(x) = \underset{x}{\operatorname{argmax}}[f(x) - C]$$

$$\underset{\theta_1,\theta_2}{\operatorname{argmax}}\, \mathbb{E}_{X,Y'}\left[\min\left(0, d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right) - \alpha\right)\right]$$

$$\min(a,b) = -\max(-a,-b)$$

$$\underset{\theta_1,\theta_2}{\operatorname{argmax}}\, \mathbb{E}_{X,Y'}\left[-\max\left(0, \alpha - d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right)\right)\right]$$

$$\underset{x}{\operatorname{argmax}}\, f(x) = \underset{x}{\operatorname{argmin}}[-f(x)]$$

$$\underset{\theta_1,\theta_2}{\operatorname{argmin}}\, \mathbb{E}_{X,Y'}\left[\max\left(0, \alpha - d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right)\right)\right]$$

ReLU

前往新目标——从回归出发

- 现在得到的两个目标
- 拉进属于同一电影的特征

$$\underset{\theta_1,\theta_2}{\operatorname{argmin}}\ \mathbb{E}_{X,Y}\left[d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right)\right]$$

- 推远属于不同电影的特征

$$\underset{\theta_1,\theta_2}{\operatorname{argmin}}\ \mathbb{E}_{X,Y'}\left[\mathrm{ReLU}\left(\alpha - d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right)\right)\right]$$

- 二者结合：Contrasive Loss

$$\mathcal{L}(X,Y) = \begin{cases} d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right), & X,Y\text{来自同一电影} \\ \mathrm{ReLU}\left(\alpha - d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right)\right), & X,Y\text{来自不同电影} \end{cases}$$

前往新目标——从回归出发

- Ref. 林思旭，Contrasive Loss

```
def create_labels(batch_size):
    labels = torch.zeros(batch_size, batch_size, dtype=torch.float32)
    for i in range(batch_size):
        labels[i, i] = 1.0
    return labels


class ContrastiveLoss(nn.Module):
    def __init__(self, margin=1.0):
        super(ContrastiveLoss, self).__init__()
        self.margin = margin
    def forward(self, output1, output2, label):
        euclidean_distance = nn.functional.pairwise_distance(output1, output2)
        loss_contrastive = torch.mean((label) * torch.pow(euclidean_distance, 2) +
                                      (1 - label) * torch.pow(torch.clamp(self.margin - euclidean_distance, min=0.0), 2))
        return loss_contrastive

contrastive_loss = ContrastiveLoss(margin=1.0)
```

前往新目标——从回归出发

- 这个策略的另一个值得商榷的地方

$$\underset{\theta_1,\theta_2}{\operatorname{argmin}}\, \mathbb{E}_{X,Y}\left[d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right)\right]$$

- 使来自同一电影的$f_{\theta_1}(X), g_{\theta_2}(Y)$距离严格为零的目标是合理的吗？

- 回顾任务：给定海报/简介，找到离它**最近的**对应简介/海报
- 目标的放宽：让来自不同电影的特征距离与来自同一电影的特征距离之差尽量大

$$\underset{\theta_1,\theta_2}{\operatorname{argmax}}\, \mathbb{E}_{X,Y,Y'}\left[d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right) - d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right)\right]$$

前往新目标——从回归出发

- 当前的目标

$$\underset{\theta_1,\theta_2}{\operatorname{argmax}}\, \mathbb{E}_{X,Y,Y'}\left[d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right) - d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right)\right]$$

- 与之前的推导完全一致，加上距离阈值α，得到损失函数
- `Triplet Margin Loss`

$$\mathcal{L}(X,Y,Y') = \mathrm{ReLU}\left[d\left(f_{\theta_1}(X), g_{\theta_2}(Y)\right) - d\left(f_{\theta_1}(X), g_{\theta_2}(Y')\right) + \alpha\right]$$

正样本　　负样本　　Margin

TripletMarginLoss

CLASS torch.nn.TripletMarginLoss(*margin=1.0, p=2.0, eps=1e-06, swap=False, size_average=None, reduce=None, reduction='mean'*) [SOURCE]

前往新目标——从回归出发

- Ref. 肖雯瑄，负采样与Triplet Margin Loss

```
class MovieLensDataset(Dataset):
    def __getitem__(self, index):
        anchor_poster = self.poster_features[index]
        positive_intro = self.intro_features[index]
        negative_index = random.choice([i for i in range(len(self.intro_features)) if i != index])
        negative_intro = self.intro_features[negative_index]
        return (anchor_poster, positive_intro), (_, negative_intro)

criterion = nn.TripletMarginLoss(margin=0.5)
for batch_idx, ((anchor_poster, positive_intro), (_, negative_intro)) in enumerate(dataloader):
        anchor_poster = anchor_poster.to(device)
        positive_intro = positive_intro.to(device)
        negative_intro = negative_intro.to(device)
        optimizer.zero_grad()
        anchor_output = model(anchor_poster, model_type='cnn')
        positive_output = model(positive_intro, model_type='bert')
        negative_output = model(negative_intro, model_type='bert')
        loss = criterion(anchor_output, positive_output, negative_output)
        loss.backward()
        optimizer.step()
```

前往新目标——从分类出发

- 分类任务的策略——交叉熵损失

$$\mathcal{L}(X, y) = -\sum_{i=1}^{N} y_i \log \frac{\exp(f(X)_i)}{\sum_{j=1}^{N} \exp\big(f(X)_j\big)}$$

- 分类在做什么？让一个样本和它对应的标签**匹配**
- 我们要做什么？把一种模态数据与另一种模态数据**匹配**

- 如果把标签也当做一种模态会发生什么？

前往新目标——从分类出发

- 如果把标签也当做一种模态会发生什么？

- 对N个标签进行one-hot编码，得到$e_1, e_2, \cdots, e_N$
- 如果输入数据X的标签是p，那么我们希望
  - $f(X)$和e_p尽可能接近　　——正样本对
  - $f(X)$和$e_{i(i\neq p)}$尽可能远离　　——负样本对
- 如何衡量$f(X)$和e_i的相似性？
  - 一个事实：$\{e_i\}_{1:N}$是N维空间上的单位正交基
  - 符合直觉的定义：$f(X)$在e_i上的投影越长，说明二者越接近→余弦相似度
- 相似性函数：$\mathrm{sim}(X, e_i) =< f(X), e_i >= f(X)_i$

前往新目标——从分类出发

- 相似性函数： $\mathrm{sim}(X, e_i) =< f(X), e_i >= f(X)_i$
- 交叉熵损失：

$$\mathcal{L}(X, y) = -\sum_{i=1}^{N} y_i \log \frac{\exp(f(X)_i)}{\sum_{j=1}^{N} \exp\big(f(X)_j\big)}$$

- 假如X的标签是p，则$y_p = 1$，$y_{i(i \neq p)} = 0$，改写损失函数

$$\mathcal{L}(X, e_1, e_2, \cdots, e_N) = -\log \frac{\exp(\mathrm{sim}(X, e_p))}{\sum_{j=1}^{N} \exp(\mathrm{sim}(X, e_j))}$$

正样本

正样本+负样本

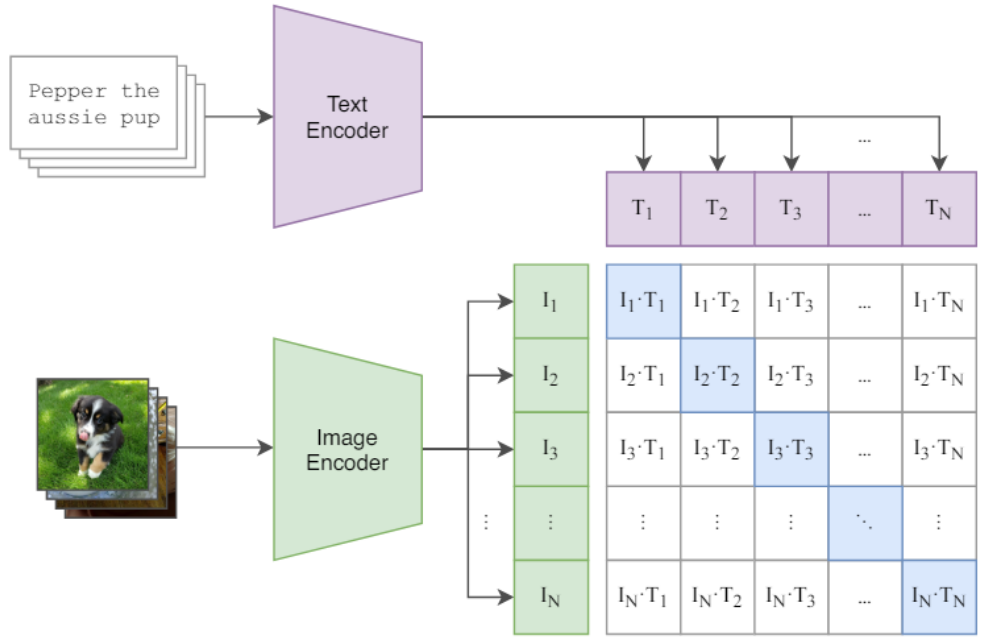
前往新目标——从分类出发

- 在$\{e_j\}_{1:N}$中寻找与X对应的标签

$$\mathcal{L}(X) = -\log \frac{\exp(\text{sim}(X, e_p))}{\sum_{j=1}^{N} \exp(\text{sim}(X, e_j))}$$

- 在$\{Y_j\}_{1:N}$中寻找与X_i对应的特征

$$\mathcal{L}(X_i) = -\log \frac{\exp(\text{sim}(X_i, Y_i))}{\sum_{j=1}^{N} \exp(\text{sim}(X_i, Y_j))}$$




```
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

前往新目标——从分类出发

- Ref. 张馨尹，CLIP模式的实现

```
def contrastive_loss(img_features, text_features, temperature=0.05):
    cosine_similarity = nn.CosineSimilarity(dim=-1)
    img_features = img_features.unsqueeze(1)
    text_features = text_features.unsqueeze(0)
    similarities = cosine_similarity(img_features, text_features)
    logits = similarities / temperature
    labels = torch.arange(len(img_features)).to(logits.device)
    loss_img = nn.CrossEntropyLoss()(logits, labels)
    loss_text = nn.CrossEntropyLoss()(logits.T, labels)
    return (loss_img + loss_text) / 2
criterion = contrastive_loss
for epoch in range(num_epochs):
    combined_model.train()
    total_loss = 0.0
    for poster, intro in train_dataloader:
        poster = poster.to(device)
        optimizer.zero_grad()
        with autocast():
            poster_out,intro_out = combined_model(poster, intro)
            loss = criterion(poster_out,intro_out)
        scaler.scale(loss).backward()
        scaler.step(optimizer)
        scaler.update()
```

Part3. 算出了ACC，然后呢？

——“人与数字有许多怪事，看看计数机里幽禁几多人质”

也许是大家最为关心的数字——ACC

- 数字的**意义**：ACC的不同取值，意味着在这个任务上做到了什么程度?
- 回顾作业中的两个数字
  - 评估指标要求top-5%
  - 训练集数据占总体数据的60%
- ACC=5%，模型没有学到任何东西，与随机赋值的结果没有区别
- ACC=20~40%，训练设计不够合理，很可能只是靠预训练模型原有参数的能力达到的效果；也可能是预训练模型微调时未固定参数导致的收敛缓慢
- ACC=60%，在训练集上完全过拟合能够达到的水平
- ACC=65~70%，建模设计较为合理时的表现，(也许是不使用预训练模型能达到的极限水平?)
- ACC=75%，几乎只能靠对适配任务的预训练模型微调达到，并且考虑到在这个任务上很容易(几乎一定会)过拟合，非训练集部分的效果实际上也只有会在40%左右
- ACC=100%，???

如何让ACC达到/逼近100%

- 作弊方法一：“我与我周旋久，宁作我”

```python
import torch.nn.functional as F
def get_dist_matrix(model, dataloader, **kwargs):
    model.eval()
    N = len(dataloader.dataset)
    distances = torch.zeros((N, N))
    with torch.no_grad():
        # Get distances of all poster-intro pair
        all_features = []
        for poster, intro in dataloader:
            poster, intro = poster.to(device), intro.to(device)
            features = model(poster, intro)
            all_features.append(features)

        all_features = torch.cat(all_features, dim=0)
        # cosine distance
        for i in range(N):
            cosine_sim = F.cosine_similarity(all_features, all_features[i].unsqueeze(0), dim=1)
            distances[i] = 1 - cosine_sim.cpu()  # 余弦距离 = 1 - 余弦相似度
            print('\r{} done'.format(i + 1), end='')

    return distances
```

$$D_{ii} = 1 - \cos\langle f_i, f_i \rangle \equiv 0$$
$$D_{ij} = 1 - \cos\langle f_i, f_j \rangle \geq 0$$

如何让ACC达到/逼近100%

- 作弊方法二：“出淤泥而不染”

```python
class IntroEncoder(nn.Module):
    def forward(self, text, image_feature):
        embedded = self.embedding(text)
        output, (hidden, cell) = self.lstm(embedded)
        image_features = image_feature.unsqueeze(1).expand(-1, text.size(1), -1)
        attn_output, attn_output_weights = self.attention(output, image_features, image_features)
        return attn_output[:, -1, :]

class CLIPModel(nn.Module):
    def forward(self, intro, poster):
        p_f = self.poster_encoder(poster)
        i_f = self.intro_encoder(intro, p_f)
        p_e = self.p_projector(p_f)
        i_e = self.i_projector(i_f)
        return i_e, p_e
```

IntroEncoder并不生产文本特征
只是学着成为图片特征的搬运工

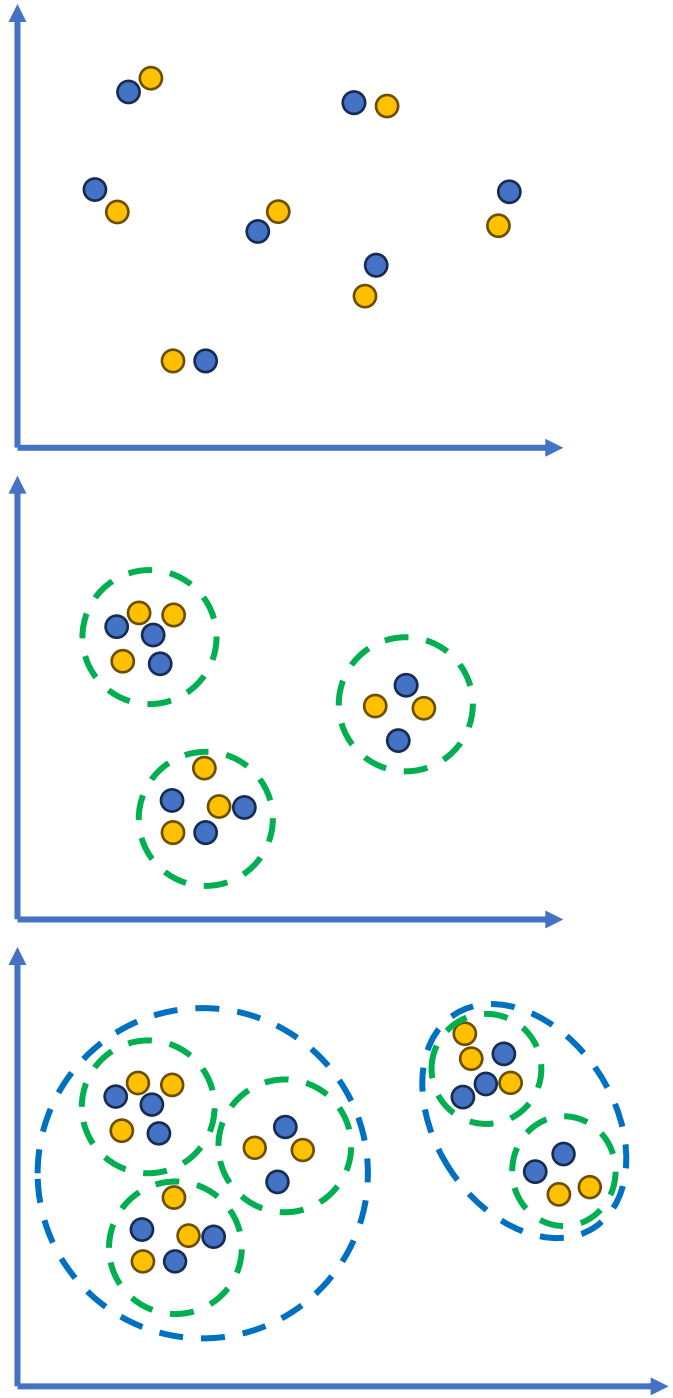
为什么难以泛化?

- 从一些数字着手看这个问题
  - MNIST: 60,000训练+10,000测试
  - CIFAR10: 50,000训练+10,000测试
  - CoCo: 330,000图像
  - ImageNet: >14,000,000图像

  - 我们的数据: ~3,000图像+文本
- 如何度量泛化性

$$\mathbb{E}_{X,y\sim\mathcal{D}_{\text{train}}}[\mathcal{L}(f(X),y)] - \mathbb{E}_{X,y\sim\mathcal{D}_{\text{all}}}[\mathcal{L}(f(X),y)]$$

- 训练数据需要
  - 足够丰富
  - 能在一定程度上反映全体数据的分布



性能的上限?

- 模型设计和训练策略并不是影响性能的全部因素
- 数据本身带来的制约——数据质量
  - 多样性、代表性、完整性、准确性、平衡性、噪声与异常……
- 我们的数据集质量如何?

1107.jpg