EI328 工程实践与科技创新实验报告

基于迁移学习的情感脑机接口分类问题

张亦昕
519030910273
上海交通大学计算机科学与工程系
jasonzhang517@sjtu.edu.cn

李智聃
519021910131
上海交通大学计算机科学与工程系
hyjrgslm-lzd@sjtu.edu.cn

摘要

关键字：情感脑机接口 迁移学习 神经网络 分类问题

1. 实验简介与背景

1.1. 背景

基于传感技术以及人工智能技术的进步，人类对于情绪的感知和认识在变得更加深刻。情绪本质上是体内物质与能量的微妙变化，随着电子感知技术的发展，越来越多的相关信息可以被利用和量化。利用可穿戴设备对于人体相关指标进行实时检测和分析，与大脑或者人体的相关状态相联系，运用分类器等技术，进行相关性的分析，可以帮助我们更好的了解自己，探知情绪的变化与波动。同时，这项技术对于人类精神疾病的预防以及治疗，预防情绪诱导的犯罪等具有显著的意义。

情感脑机接口 (Affective Brain-Computer Interfaces) 是一种对人的情绪进行识别或调控的脑机接口。通过相关外设传感感知如脑电、眼动等信号，之后进行情绪解码，可以感知目标的情绪状态（如效价、唤醒度等指标），结合刺激控制器等设备，可以对目标的情绪进行相关的调节。上海交通大学吕宝粮教授团队从 2008 年开始进行相关的研究，目前已经在相关顶级学术会议上发表了多篇学术论文，取得了卓越的成果，且已经投入了临床试验。结合相关的医疗临床资源，情感脑机接口在未来将会有非常广阔的发展空间与开发价值。

1.2. 实验简介

利用情感脑机接口相关技术采集到的数据，基于脑电的情绪识别就成为了一个典型的模式分类问题。不同于传统的图像和语音信号，脑电信号具有非平稳特性，信噪比低以及个体差异性大的特点，因此数据具有多维度，波动大的特征。本实验从应用的角度出发，利用迁移学习相关算法构建具有跨被试能力、普适性强的情绪识别模型，利用已知多名被试的脑电情绪数据训练模型，尝试对未知被试的脑电数据进行情绪识别，并优化其性能与准确率。

1.3. 实验数据描述

本实验用到的数据是数据集是上海交大计算机系类脑计算与机器智能研究中心 (BCMI) 的脑电情感数据集（SJTU Emotion EEG Dataset，简写为 SEED）的一个子集。实验选取了 3 类情绪（高兴、悲伤、中性）共 15 段电影片段作为情绪诱发刺激素材。每段视频时长约三分钟，每类情绪有 5 段电影片段。15 名被试每名被试均在实验过程中观看 15 段视频素材，以诱发相应情绪。

本实验所用的数据集分为 15 组，对应 15 名被试者。每一组的数据输入 (X) 为一个 3394×310 的矩阵，对应实验中 3394 次采样，每一次采样的特征维度为 310 维。每组的数据标签 (Y) 维一个 3394×1 的向量，对应实验中每一次的采样的情绪状态，数

据标号为三类情绪：平静、悲伤和高兴，对应 0、-1、1。数据存储方式为 mat 文件。

1.4. 实验要求

对于 15 名被试者的数据，将其中 14 名被试者的数据进行拼接，在剩余 1 名被试者的数据上进行测试，训练支持向量机 (Support Vector Machine，简写为 SVM) 作 15 折交叉验证，以其平均准确率为基准线，准备进行后续提升。

按照相同的数据划分方式，对于上述数据进行迁移学习，减小分类器因为个体差异而产生的影响，获得更准确的分类器，提高测试集平均准确率。

2. 算法思想与程序设计分析

在本实验中，本小组使用了多种机器学习的算法与网络结构，接下来对于它们进行简单的介绍与分析，结合小组在本实验中的实际应用，具体说明用到的算法和原理。

2.1. SVM 支持向量机

2.1.1 方法介绍

支持向量机 (Support Vector Machine，简写为 SVM)，是一种有监督的机器学习模型。其作用是使用分类算法来解决分类问题。为每个类别提供一组带有标签的输入数据之后，SVM 就能够有一定的对于新输入数据组的分类能力。与神经网络等方法相比，SVM 有两个主要优点：更快的速度以及对于文本分类问题具有更高的准确率，因此它在本实验的情景中更加符合。

在一个高维空间里，假设我们想解决的分类问题是一个线性可分问题，我们想要设法寻找一个超平面，使之基本能够分开标签不同的两组数据。同时，根据数据点与这个超平面的距离，我们要设法寻找一个最理想的平面，它可以最大化到两组最近标签数据元素的距离，这样训练出的分类模型具有最好的鲁棒性。

$$r = \frac{W^T X_i + b}{||W||}$$

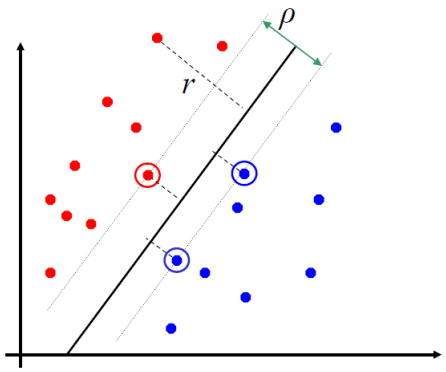

图 1. Principles of SVM

经过转化，SVM 求解的就是一个最优化问题：

$$\min ||W||^2 \ s.t. y_i(W^T x_i + b) \geq 1$$

通过求偏导等方式，可以进行不断优化，最终求解出最优的超平面。

即使是对于无法用超平面划分的数据，可以通过核函数将样本映射到高维空间中，这样就变回了线性可分的超平面问题。

2.1.2 使用 SVM 支持向量机的基准分类模式

实验中，本小组使用了 sklearn 的 SVM 进行训练和测试，参数均选用默认参数，决策函数形状参数选用 ovo，即一对一模型。将数据拼接之后，每次使用剩余的一组数据进行测试 (svm 的 predict 函数)。使用一个数组列表保存对于每一组测试集的准确率。

```
precision = []
for i in range(scale_num):
    x = X[i]
    y = Y[i]
    clf = svm.SVC(
        decision_function_shape='ovo')
    clf.fit(x, y)
    p = output_precision(clf, eeg_x[i],
        eeg_y[i])
    precision.append(p)
```

2.2. 迁移学习介绍

当前的人工智能技术大多需要大量的高质量数据来作为基础。但是获得足够的优质数据存在很大

的困难。因此，我们希望可以通过已知的数据（已分类）训练出的模型，对未知的数据（未分类）做出准确的预测。在这次大作业中，每个人的 EEG 信号存在个体差异。我们希望可以找到一种迁移学习方式，可以在对一部分人的 EEG 信号进行训练后，能够对另一部分 EEG 信号做出较准确的分类。

2.3. 迁移成分分析 (TCA) 的数据降维法

2.3.1 方法介绍

TCA 是一种很朴素的迁移方法。[5] TCA 假设，源于和目标域的边缘分布不同，也就是 $P(X_s) \neq P(X_T)$。因此，传统的机器学习方法效果很差（见 SVM 分类器效果）。但是，假设存在一个映射 ϕ，使得 $P(\phi(X_s)) \approx P(\phi(X_t))$，更进一步，有 $P(Y_s\ \phi(X_s)) \approx P(Y_t|\phi(X_t))$。那么，我们可以找到这个映射，对 X_s 和 X_t 映射后，再利用传统机器学习方法进行训练和分类。

2.3.2 数学定义

TCA 利用最大均值差异（MMD）作为源域和目标域的距离，其定义为

$$dist(X'_{src}, X'_{tar}) = \left\| \frac{1}{n_1}\sum_{i=1}^{n_1}\phi(x_{src_i}) - \frac{1}{n_2}\sum_{i=1}^{n_2}\phi(x_{tar_i}) \right\|_{\mathcal{H}}$$

为了求解迁移映射，构建核函数：

$$K = \begin{bmatrix} K_{src,src} & K_{src,tar} \\ K_{tar,src} & K_{tar,tar} \end{bmatrix}$$

以及 L：

$$L_{ij} = \begin{cases} \frac{1}{n_1^2} & x_i, x_j \in X_{src} \\ \frac{1}{n_2^2} & x_i, x_j \in X_{tar} \\ -\frac{1}{n_1 n_2} & \text{otherwise} \end{cases} \quad (1)$$

这样，求解 MMD 距离就变为：

$$trace(KL) - \lambda trace(K)$$

可以通过降维的方式加速运算：

$$\tilde{K} = (KK^{-\frac{1}{2}}\tilde{W})(\tilde{W}^T K^{-\frac{1}{2}}K) = KWW^TK$$

那么，最后优化的目标就得到了：

$$\min_W tr(W^TKLKW) + \mu tr(W^TW) s.t. W^TKHKW = I_m$$

其中，$H = I_{n_1+n_2} - \frac{1}{(n_1+n_2)}11^T$

最后，我们只需要求解前 m 个特征值 [3]，就可以得到我们需要的以后的矩阵 W。

2.3.3 算法实现

对于之前的数学推导计算，用 python 进行实现：

```
class TCA:
def __init__(self, dim=30, lamb=1,
    gamma=1):
     self.dim = dim
     self.lamb = lamb
     self.gamma = gamma

def fit(self, Xs, Xt):
     '''
     :param Xs: 数据数 * 每个数据的特征
         数，源特征
     :param Xt: 数据数 * 每个数据的特征
         数，目标特征
     :return: 二者经过TCA之后的Xs_new和
         Xt_new
     '''

     X = np.hstack((Xs.T, Xt.T))
     X /= np.linalg.norm(X, axis=0)
     m, n = X.shape
     ns, nt = len(Xs), len(Xt)
     e = np.vstack((1 / ns * np.ones((ns
         , 1)), -1 / nt * np.ones((nt, 1)
         )))
     M = e * e.T
     M = M / np.linalg.norm(M, 'fro')
     H = np.eye(n) - 1 / n * np.ones((n,
          n))
     K = kernel(X, None, gamma=self.
         gamma)
     n_eye = n
     a, b = K @ M @ K.T + self.lamb * np
         .eye(n_eye), K @ H @ K.T
```

```
        print('start w/V 计算')
        w, V = scipy.linalg.eig(a, b,
            overwrite_a=True, overwrite_b=
            True, check_finite=False)
        ind = np.argsort(w)
        A = V[:, ind[:self.dim]]
        Z = A.T @ K
        Z /= np.linalg.norm(Z, axis=0)

        Xs_new, Xt_new = Z[:, :ns].T, Z[:,
            ns:].T
        Xs_new = np.dot(Xs_new, np.eye(
            Xs_new.shape[1])).astype(float)
        Xt_new = np.dot(Xt_new, np.eye(
            Xt_new.shape[1])).astype(float)
        print('TCA完成')
        return Xs_new, Xt_new
```

由于设备性能原因，我们随机选择了 3394 条源域数据与 3394 条目标域数据进行迁移学习。但是，由于训练数据相对较少，计算的效率比较低。而且由于数据量少且特征不够明晰，降维后的数据不能够体现原本数据的特征。生成后的网络分类效果不好。TCA 迁移学习并没有取得较好的结果，准确率较 SVM 相比并无明显的提升。因此，我们决定转为使用域对抗网络 (DANN) 进行迁移学习。

2.4. 基于反向传播的无监督域迁移 (Unsupervised Domain Adaptation by Backpropagation, UDAB)

2.4.1 方法介绍

域适应是迁移学习的一个分支，目的是构造一种映射，将具有不同分布的源域和目标域映射到同一个特征空间，使他们在映射后具有相似的分布。这样，就可以用一个带有标签的数据集训练分类器，来对没有标签的目标域数据进行较高精确度的分类。

生成对抗网络（GAN）是 GoodFellow 等人在 2014 年提出的一种网络，结构如图 2 所示：

GAN 中包含一个生成器和一个判别器。生成器负责生成假数据，判别器则区分输入的数据是真数据还是假数据。生成器希望生成的数据可以骗过判别器，判别器希望可以减少被骗。二者博弈，最后系统能够达到稳定状态。

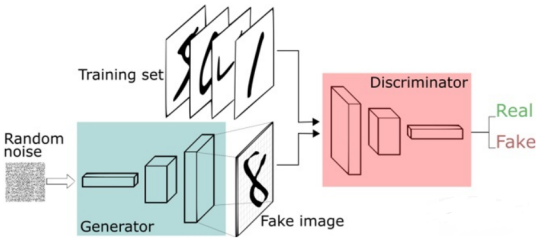

图 2. Generating Adversarial Network

GAN 的思想可以应用到域迁移中。[2] 我们将目标域的数据看做生成器生成的数据，此时，生成器的目标是从源域和目标域中提取特征，使判别器无法区分提取的特征来自目标域还是源域。我们希望挑选出的特征有如下两个特征：

一、通过这些特征，你无法区分它来自源域还是目标域

二、通过这些特征，你可以有效地完成分类

也就是说，域对抗迁移的网络损失由两部分构成：训练损失（标签预测损失）和域判别损失。这就是域对抗网络的基本思想 [6]。DANN 网络的结构如图 3 所示：

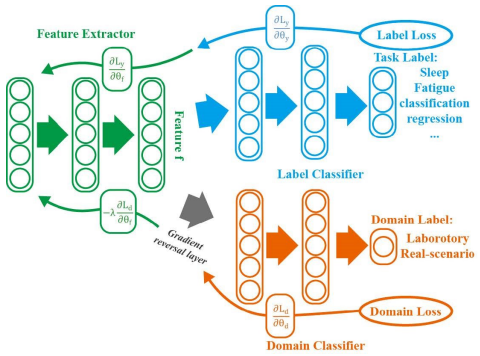

图 3. Domain Adversarial Neural Networks

DANN 结构主要为三个部分：特征提取器（绿色），将数据映射到特定的特征空间，目的是使标签预测器可以分辨出数据的分类，但是域判别器不能分辨数据的来源。

标签预测器（蓝色），对输入的数据进行分类，尽可能得到正确的分类结果。

域判别器（红色），对输入的数据，判断其来自

哪个域。

其中，特征提取器和标签分类器构成了一个前馈神经网络，而在特征提取器后，通过一个梯度反转层（GRL）连接域判别器 [1]。在训练过程中，对来自源域的数据，网络最小化标签预测器的损失；对来自源域和目标域的数据，网络最小化域判别器的损失。

基于以上的理论基础，可以在 DANN 模型的基础上提出在本次大作业中使用的方法：基于反向传播的无监督域迁移（UDAB）。

2.4.2 数学模型

假设模型适用输入样本 $x \in X$，其中 X 是输入空间，标签 y 属于标签空间 Y，假设 $Y = \{1, 2, \ldots, L\}$ 是一个有限集。假设在 $X \otimes Y$ 上存在两个分布 $\mathcal{S}(x, y)$ 和 $\mathcal{T}(x, y)$，称为源分布与目标分布。

接下来，定义一个预测标签与所属域的前馈网络。网络由三部分组成：第一部分是一个 D 维的特征提取器 G_f，产生的特征向量 $f \in \mathbb{R}^D$，这一层的参数集合记作 θ_f。特征 f 被送入标签预测器 G_y 与域判别器 G_d，对应参数集合分别为 θ_y,θ_d

我们要寻找源域与目标域的不变性，就是要让分布 $S(f) = \{G_f(x; \theta_f) | x \sim S(x)\}$ 与 $T(f) = \{G_f(x; \theta_f) | x \sim T(x)\}$ 相似。为了找到领域不变特征，我们要寻找能够最大化域判别器的 θ_f，同时最小化标签预测器与域判别器的损失。更一般地说，有下式：

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) &= \sum_{\substack{i=1\ldots N \\ d_i=0}} L_y(G_y(G_f(x_i; \theta_f); \theta_y), y_i) \\ &\quad - \lambda \sum_{i=1\ldots N} L_d(G_d(G_f(x_i; \theta_f); \theta_d), y_i) \\ &= \sum_{i=1\ldots N \ d_i=0} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1\ldots N} L_d^i(\theta_f, \theta_d) \end{aligned} \quad (2)$$

其中，$L_y(\cdot, \cdot)$ 是标签预测器的损失函数，$L_d(\cdot, \cdot)$ 是域判别器的损失，L_y^i 和 L_d^i 表示第 i 个标签的损失值。参数 λ 是两个学习目标之间的权重。我们的目标是找到参数 $\hat{\theta}_f$,$\hat{\theta}_y$,$\hat{\theta}_d$ 可以最小化上式，即

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d)$$

$$\hat{\theta}_d = \arg\min_{\theta_d} E(\theta_f, \theta_y, \hat{\theta}_d)$$

在反向传播时，根据如下式子更新参数:

$$\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right) \quad (3)$$

$$\theta_y \leftarrow \theta_y - \mu \frac{\partial L_d^i}{\partial \theta_f} \quad (4)$$

$$\theta_d \leftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d} \quad (5)$$

其中，μ 是学习率。参数 λ 将其与 SGD 算法区分开来。这个参数可以看做伪函数 $R_\lambda(x)$:

$$R_\lambda(x) = x \quad (6)$$

$$\frac{dR_\lambda}{dx} = -\lambda \mathrm{I} \quad (7)$$

最终得到需要优化的式子:

$$\begin{aligned} \tilde{E}(\theta_f, \theta_y, \theta_d) &= \sum_{\substack{i=1\ldots N \\ d_i=0}} L_y(G_y(G_f(x_i; \theta_f); \theta_y), y_i) \\ &\quad + \sum_{i=1\ldots N} L_d(G_d(R_\lambda(x_i; \theta_f); \theta_d), y_i) \end{aligned} \quad (8)$$

通过 (3)-(5)，可以实现对 (8) 做 SGD，学习后可以用标签预测因子 $y(x) = G_y(G_f(x; \theta_f); \theta_y)$ 来预测标签。

2.4.3 代码实现

我们使用 pytorch [4] 实现上述模型。该模型有三个子网络：特征提取器、标签预测器与域判别器。我们使用交叉熵损失与 Adam optimizer 来进行模型优化。

特征提取器：首先有一个 310 维到 256 维的线性网络层，之后用一个批标准化层 (BatchNorm1d)，然后设置一个线性整流层。再用 256 维到 128 维的

线性网络层，批标准化层与线性整流层与之前相同，最后加一个 Dropout 层，避免模型过拟合，增强模型的泛化能力。输入为 310 维的数据，输出为 128 维。

标签预测器：接受 128 维的输入，有两组线性网络层、批标准化层和线性整流层，将数据从 128 维映射到 64 维最后到 32 维，最后用线性网络层映射为 3 维数据，并附加 softmax 进行归一化。

域判别器：接受 128 维的输入，有一组线性网络层、批标准化层和线性整流层，将数据从 128 维映射到 32 维，最后用线性网络层映射为 2 维数据，并附加 softmax 进行归一化。

```
class DANN(nn.Module):
#定义网络的形状
def __init__(self, track_running_stats,
    momentum):
    super().__init__()
    self.feature_extractor = nn.
        Sequential(
        nn.Linear(310, 256),
        nn.BatchNorm1d(num_features
            =256, momentum=momentum,
            track_running_stats=
            track_running_stats),
        nn.ReLU(),
        nn.Linear(256, 128),
        nn.BatchNorm1d(num_features
            =128, momentum=momentum,
            track_running_stats=
            track_running_stats),
        nn.Dropout(),
        nn.ReLU()
    )
    self.class_classifier = nn.
        Sequential(
        nn.Linear(128, 64),
        nn.BatchNorm1d(num_features=64,
            momentum=momentum,
            track_running_stats=
            track_running_stats),
        nn.ReLU(),
        nn.Dropout(),
        nn.Linear(64, 32),
        nn.BatchNorm1d(num_features=32,
            momentum=momentum,
            track_running_stats=
            track_running_stats),
        nn.ReLU(),
        nn.Linear(32, 3),
        nn.LogSoftmax(dim=1)
    )
    self.domain_classifier = nn.
        Sequential(
        nn.Linear(128, 32),
        nn.BatchNorm1d(32, momentum=
            momentum,
            track_running_stats=
            track_running_stats),
        nn.ReLU(),
        nn.Linear(32, 2),
        nn.LogSoftmax(dim=1)
    )
#定义传播函数
def forward(self, input_data, alpha=0):
    feature = self.feature_extractor(
        input_data)
    reverse_feature = ReverseLayer.
        apply(feature, alpha)
    class_pred = self.class_classifier(
        feature)

    domain_pred = self.
        domain_classifier(
        reverse_feature)

    return class_pred, domain_pred
```

反向层：在特征提取器向域判别器传播时，数据被原样传播。但是域判别器进行梯度反向传播的时候，应乘一个负的系数，减小特定数据带来的影响。

```
class ReverseLayer(torch.autograd.
    Function):

@staticmethod
def forward(ctx, x, alpha):
    ctx.alpha = alpha
    return x.view_as(x)
```

```
    @staticmethod
    def backward(ctx, grad_outputs):
        output = grad_outputs.neg() * ctx.
            alpha
        return output, None
```

与 SVM 实验类似，我们进行了 15 折交叉验证，依次选取一个参与者的数据作为没有标签的数据，通过与 SVM 实验的对比，可以得出域适应的贡献。

3. 实验结果分析

实验结果：将训练好的网络作为参数，代入测试集的输入 X 向量，得到一组预测的标签值，与实际数据的标签相对比，可以计算得到准确率。

3.1. 基准：SVM 分类器

基础的支持向量机对于数据集的分类已经有比较高的分类准确率。对于 15 组数据总的准确率：61.23%

其中，对于第 0 组数据的分类效果最差，准确率仅为 32.53%，对于第 12 组数据的分类效果最好，准确率为 91.31%。

运行时间方面，每做一组测试约用时 2 到 3 分钟，15 组数据总用时 30 分钟左右。

3.2. 迁移成分分析 (TCA) 的数据降维法

由于该方法要求取矩阵的特征值，而数据的规模比较大，所以本方法的计算时间和内存占用率都很高（每做一组测试需要用时 1 小时左右，16G 内存占用 70% 左右），本小组只能多次基于不同组数据进行降维训练之后测试相同的一组，由于数据降维使数据丢失了许多本来的特征，形成的网络分类能力比较低，结果不是很理想，最终该方法被舍弃。

3.3. 基于反向传播的无监督域迁移 (UDAB)

对于 15 组数据总的准确率 76.33%

其中，对于第 3 组数据的分类效果最好，准确率为 93.13%，对于第 4 组数据的分类效果最差，准确率仅为 50.74%。

使用算法	SVM	TCA	UDAB
总体数据准确率	61.23%	32.27%	76.33%
用时	30 分钟	1 小时 (每组)	15 分钟

表 1. 不同算法的准确率对比

运行时间方面，由于使用了 GPU 进行训练和计算，整体的速度提升了非常多，全部数据训练完毕需要 15 分钟左右。

4. 结论与致谢

4.1. 实验结论

本次实验中，本小组通过不同的深度学习框架尝试了训练支持向量机与迁移学习，以不断提高对于分类问题的准确率。在不同的方法中，SVM 是一个对于文本分类问题高效高准确率的一种训练方式，在本次实验中颇为有效。

TCA 是一种基于统计信息改变映射函数的一种分类方式，但是矩阵运算较为复杂，因此效率较低。同时因为数据数量少和数据维度降幅过大，迁移学习准确率较低。受制于设备性能的原因，本小组未能做进一步的尝试，但是它仍然在特定问题上具有较好的表现。

UDAB 利用了神经网络进行训练和调整。域判别器用于消除数据个体带来的差异，是一种非常有效的迁移学习算法，对于小规模分类问题是一种非常有效的分类方式。同时，借助 CUDA 在 GPU 上高速运行，训练的时间被显著降低。两者结合，能够以较低的时间成本获得理想的实验效果。

4.2. 致谢

感谢上海交通大学电院计算机系给我们开设了这么高质量的课程，让我们系统学习了人工智能和深度学习方面的理论知识，全面的接触了各种算法和学习框架。同时接触到了最前沿的实际应用，看到了这个学科领域最先进的技术和极为广阔的前景。同时，提高了我们的论文阅读和学习能力，本次实验接触到的迁移学习是对于我们非常陌生的概念，相关的算法和技术需要我们查找相关资料、阅读论文进行学习和分析，对我们的能力是很大的提升。

同时感谢疫情非常时期继续坚持授课和答疑的吕宝粮教授和助教们，在艰难的环境下与我们共同克服疫情带来的影响，坚持对于知识和技术的追求，对于我们来说也是一种卓越的鼓舞。

4.3. 小组分工

张亦昕：论文阅读，baseline 和 UDAB 代码部分，报告 TCA 以外内容和 latex 整合。

李智聃：论文阅读，TCA 和 UDAB 代码部分，报告的 TCA、UDAB 部分。

参考文献

[1] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis Bach and David Blei, editors, Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pages 1180–1189, Lille, France, 07–09 Jul 2015. PMLR. 5

[2] Yun Luo, Li-Zhen Zhu, Zi-Yu Wan, and Bao-Liang Lu. Data augmentation for enhancing eeg-based emotion recognition with deep generative models. Journal of Neural Engineering, 2020. 4

[3] Python Official. Numpy turorial. https://numpy.org/doc. 3

[4] Python Official. Pytorch turorial. https://pytorch.org/docs/stable/index.html. 5

[5] Wei-Long Zheng and Bao-Liang Lu. Personalizing eeg-based affective models with transfer learning. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, page 2732–2738. AAAI Press, 2016. 3

[6] 王晋东不在家. 小王爱迁移. https://zhuanlan.zhihu.com/p/130244395. 4