天津大学

《数据挖掘》课程报告

基于随机森林算法的 XGB 和 LGB 分类器解决电动车分类问题

学　　号 3021001526

姓　　名 张铨

邮　　箱 7176656@qq.com

学　　院 智能与计算学部

专　　业 人工智能

年　　级 2021

任课教师 王熠

2023 年 11 月 27 日

摘要

本课题来自2021级人工智能23241课程《数据挖掘》的结课大作业，目的是应用异常检测，数据增强，数据挖掘方法实现电动车分类。

数据处理部分，对所有数据进行统一打包成csv格式，使用线性插值算法补全特征数不足的数据，使用DBSCAN聚类算法进行异常检测，排除21个离群点。

分类方法采用xgb，lgbm，cat，神经网络等多种方法进行试验，发现随机森林算法具有举足轻重的作用，其稳定性极高，鲁棒性和精确性佳，在所有基于RF的树状算法中都有非常好的表现，神经网络算法和树状算法都能取得很好的效果。

结果为，最佳的精确度acc来自xgb-rf和lgbm-rf算法，为0.9934，在评测网站测试后正常发挥，完美解决电动车分类问题。

关键词：分类；XGBoost；CatBoost；LightGBM；GBDT；RandomForest；异常检测；DBSCAN；神经网络；DNN；CNN；RNN；

目录

1 课题介绍

1.1 课题背景

本课题来自2021级人工智能23241课程《数据挖掘》的结课大作业，目的是应用异常检测，数据增强，数据挖掘方法实现电动车分类。数据标签分为电动车充电（1）与非电动车充电（0），数据特征为280维的频率谱。

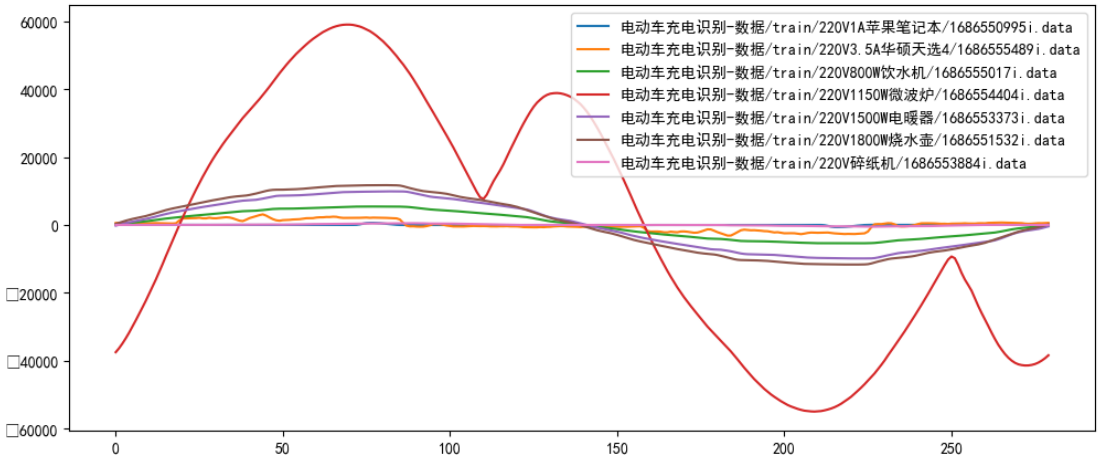
1.2 总体研究方案

总体实现高精度分类即可，首先对数据进行EDA观测特征，然后打包成csv文件，读入模型进行训练，最后预测，提交到网站观察正确率等相关指标。

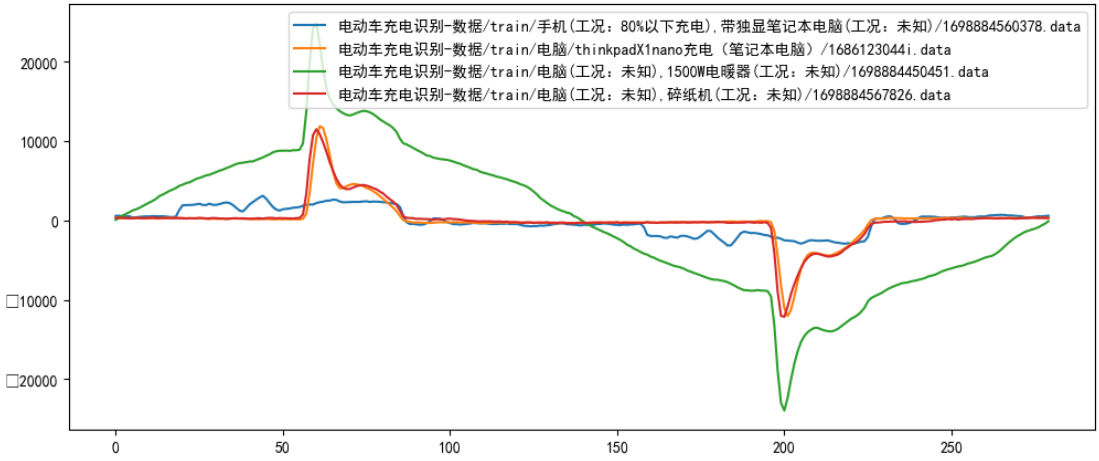
2 数据统计特性分析及可视化

2.1 可视化

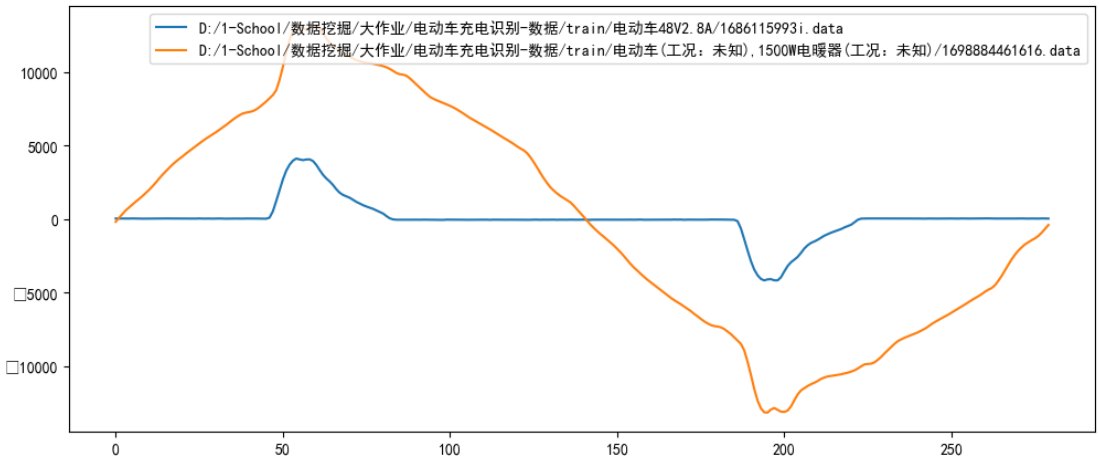
首先对数据进行EDA，观察到都是长度为280的正弦波形，形状不一，比如微波炉这个就是两个波形叠加在一起导致的。



但是叠加态的波形往往比较逆天，残次不齐，感觉是采样的数据中，电脑的波形把所有人都给带偏了。



再来看一下电动车是什么样的。



看来电动车的也是类似的，没什么不同的。可以粗略的认为所有数据都遵循类似的规律，是一个正弦波形。[ZZW+23]

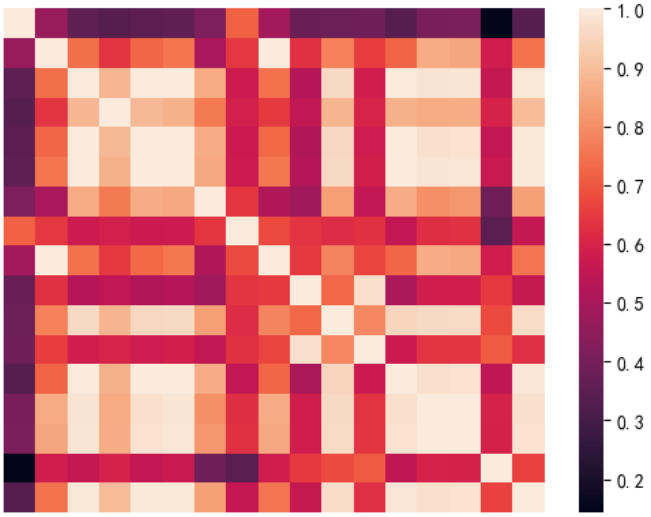
到此为止我们已经不难发现，这些数据基本上都是使用模板波形进行叠加而来的数据，我们可以采集一套模板波形进行拼凑，基本可以得出所有所需的标签，但鉴于这个课是数据挖掘课，不是模板匹配课，我们还是使用正常的机器学习方法解决问题。

2.2 统计特性分析

首先第一步，算一下分位数，std和mean，偏锋度。

	count	mean	std	min	25%	50%	75%	max	skew	kurt	name
0	280.0	-1.482143	130.360294	-656.0	-20.0	-0.5	20.0	611.0	-0.372662	14.834345	220V1A苹果笔记本
1	280.0	-52.396429	1514.020221	-3140.0	-694.0	-132.0	664.5	3111.0	-0.086289	-0.866044	220V3.5A华硕天选4
2	280.0	-0.3	3823.2574	-5435.0	-3775.0	-6.0	3782.0	5430.0	-0.000852	-1.51371	220V800W饮水机
3	280.0	-17.332143	36765.823673	-55011.0	-34988.5	-1063.0	33927.75	59040.0	0.060809	-1.415367	220V1150W微波炉
4	280.0	-0.207143	6949.66486	-9891.0	-6905.5	-10.5	6890.75	9890.0	-0.000042	-1.523618	220V1500W电暖器
5	280.0	-0.925	8249.822327	-11685.0	-8082.25	34.0	8098.5	11737.0	0.000522	-1.51668	220V1800W烧水壶
6	280.0	-1.364286	241.593907	-528.0	-93.25	0.0	90.5	528.0	0.000958	-0.050353	220V碎纸机
7	280.0	0.539286	89.121108	-259.0	-8.0	0.0	8.0	259.0	0.094551	3.629248	华为P30PRO手机
8	280.0	-53.110714	1574.966987	-3152.0	-703.0	-140.5	671.25	3114.0	-0.094837	-0.822718	手机(工况：80%以下充电),带独显笔记本电脑(工况：未知)
9	280.0	2.082143	2589.627956	-12038.0	-266.0	41.5	291.0	11878.0	-0.012781	9.081365	thinkpadX1nano充电（笔记本电脑）
10	280.0	-11.571429	8802.693249	-23950.0	-6868.75	-6.0	6833.0	24805.0	0.006019	-0.269439	电脑(工况：未知),1500W电暖器(工况：未知)
11	280.0	6.453571	2707.811064	-12133.0	-264.5	0.0	292.0	11499.0	-0.121523	7.636418	电脑(工况：未知),碎纸机(工况：未知)
12	280.0	-0.232143	196.134526	-277.0	-201.0	1.0	200.5	274.0	-0.002868	-1.561772	电风扇3挡
13	280.0	-53.482143	5237.096264	-8385.0	-4888.5	28.5	4784.75	7913.0	-0.028794	-1.302867	电风扇(工况：未知),800W饮水机(工况：烧水),带独显笔记本电脑(工况：未知)
14	280.0	-54.846429	5432.82977	-8813.0	-5171.25	28.5	5114.0	8273.0	-0.029237	-1.297935	电风扇(工况：未知),带独显笔记本电脑(工况：未知),碎纸机(工况：未知),800W饮水机(...
15	280.0	0.585714	1281.303494	-4165.0	-40.0	6.0	44.0	4121.0	-0.036531	4.75871	电动车48V2.8A
16	280.0	-1.282143	7720.254567	-13183.0	-6883.25	-9.0	6889.5	13056.0	-0.000634	-1.291262	电动车(工况：未知),1500W电暖器(工况：未知)

统计量几乎是看不出任何抓手，接下来看相似度。



这一看不得了，pearson相似度竟然有这么高的，这就说明很多情况下不同的电器是有线性相关性的。

3 数据预处理

3.1 数据集成与转换

使用python os.walk函数遍历文件夹，读取数据，打包成一个具有280维特征的数据集，并附上label。

```
for dir in dirsyes:
    for path, dir, files in os.walk(dir):
        print(len(files))
        for file in files:
            realpath = os.path.join(path, file)
            raw_data = pd.read_csv(realpath, header=None, names=['data']).to_numpy()

            raw_data = [1] + list(raw_data.flatten())
            dataframe_datas.loc[len(dataframe_datas)] = raw_data
```

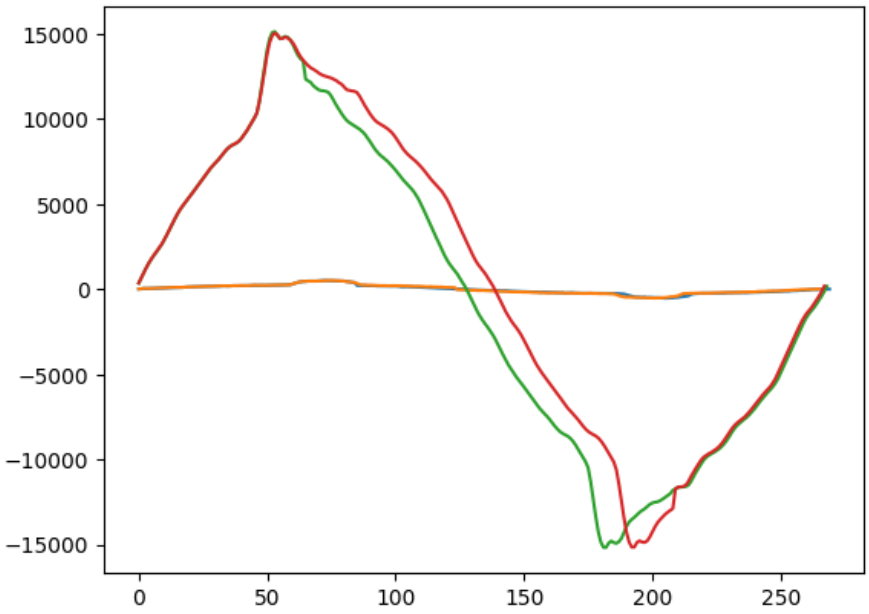
对于后续神经网络的训练，不能像树状模型那么粗糙，必须进行归一化，此处选用的是MinMax归一化，主要就是为了保存波形的形状信息，尽可能地无损传给神经网络。

```
def scale(x):
    x = np.array(x)
    xmean = x.mean()
    xstd = x.std()
    xmin = x.min()
    xmax = x.max()
    # x = (x - xmean) / xstd
    x = (x - xmin) / (xmax - xmin)
    return x
```

3.2 异常检测与异常数据处理

在打包过程中，发现有的数据不足280，一开始直接drop了，后来测试的时候发现，有的test数据也不足280，这就不能drop了，会出问题。

为了更精确的处理这些数据，还是先EDA一下看看情况。

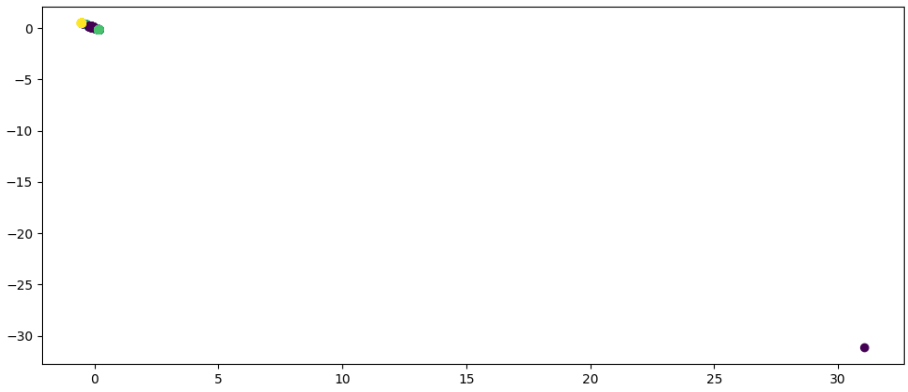


可以看出损失的长度不大，大部分波形都保留完好，但是尽量为了避免对数据整体产生较大的影响，此处使用线性插值法，补齐所有异常数据点。

```
if len(raw_data) != 280:
    x = np.linspace(0, 1, len(raw_data))
    y = raw_data.flatten()
    f = interpolate.interp1d(x, y, kind="quadratic")
    xnew = np.linspace(0, 1, 280)
    ynew = f(xnew)
    raw_data = ynew.reshape((-1, 1))
```

4 异常检测及分析

使用DBSCAN算法，对数据进行minmax标准化后进行聚类分析，找到离群点并剔除。[KGJV83]



显然是有离群点的，具体统计后有21个离群点，标签为-1，将其剔除后进入训练部分。定量分析数据表如下：

表 1: DBSCAN

Metric	Score
Homogeneity	0.938
Completeness	0.272
V-measure	0.421
Adjusted Rand Index	0.231
Adjusted Mutual Information	0.419

5 分类模型设置与训练

5.1 XGBoost

首先上场的是XGB比赛神器，不需要太多的操作，直接塞进去raw文件，反正树类模型不太吃归一化，稍微离谱一点的数据，保证纯净的生数据，让DT更好决策。[HZRS16]

5.1.1 XGBRFClassifier

使用python xgboost模块XGBRFClassifier，maxdepth=5, estimators=140, 训练集：测试集 = 3：1

表 2: XGBoost

Metric	ACC	F1score	Recall	precision
round 1	1	1	1	1
round 2	0.9961	0.9905	1	0.9811
round 3	0.9902	0.9630	1	0.9286

震惊，过于炸裂，表现极好，现在放到测评网站上看看分数。

图 1: XGBRFClassifier评测网站

5.1.2 XGBClassifier

非常奇怪的是，使用普通的xgb，即使用dt作为分类器的情况下，非常容易陷入全0或者全1，即使分数体现上仍然很高。

5.2 LightGBM

LightGBM是谷歌开发的一种流式训练器，以快为著称，实验的时候发现了一些奇怪的细节。首先是常规的三次对照试验，参数随便写的，效果如下。[HLVDMW17]

表 3: LGBMClassifier

Metric	ACC	F1score	Recall	precision
round 1	0.9706	0.8966	1	0.8125
round 2	0.9902	0.9714	1	0.9444
round 3	0.9804	0.9444	1	0.8947

效果好像不如XGBRFClassifier，但是仍然需要特别注意的一个参数叫boosting type，有三种选项，此处采用的是rf，即随机森林分类器，如果改成gbdt，那么又会陷入XGBClassifier的全0全1中，这有关于算法的设计问题。

3021001526	LGBM RF	0.9934	0.9901	1	0.995

图 2: LightGBM评测网站

5.3 CatBoost

与 XGBoost 和 LightGBM 不同，CatBoost 构建对称（平衡）树。在每一步中，前一棵树的叶子都使用相同的条件进行拆分。选择损失最低的特征分割对并将其用于所有级别的节点。这种平衡的树结构有助于高效的 CPU 实现，减少预测时间，模型结构可作为正则化以防止过度拟合。[KSH12]

iteration=100，三次测试，稳定性非常恐怖。

表 4: CatBoostClassifier

Metric	ACC	F1score	Recall	precision
round 1	1	1	1	1
round 2	1	1	1	1
round 3	1	1	1	1

预测电动车时，无一例外全0，稳定全0，效果极差。

5.4 RandomForest

到底是什么样的魔力让随机森林如此出色呢，此处单独把他捡出来对比一下。使用python xgboost模块RandomForestClassifier，maxdepth=5, estimators=140, 训练集：测试集 = 3：1

表 5: RandomForestClassifier

Metric	ACC	F1score	Recall	precision
round 1	1	1	1	1
round 2	1	1	1	1
round 3	1	1	1	1

天哪，太可怕了，评测网站的得分也非常惊人。

图 3: RandomForestClassifier评测网站

5.5 GBDT

难道真的是GBDT的锅吗？[SKK18]此处使用GradientBoostingClassifier进行试验，maxdepth=5, estimators=140, 得出的本地测试结果如下：

表 6: GradientBoostingClassifier

Metric	ACC	F1score	Recall	precision
round 1	1	1	1	1
round 2	1	1	1	1
round 3	1	1	1	1

本地效果很好，网站效果也很好，稳定得到99的ACC。这说明GBDT没有问题，是XGB和LGB等继承算法的策略有问题。

gbdt	gbdt	0.9868	0.99	0.99	0.99

图 4: GBDTClassifier评测网站

5.6 DNN

树状模型全都结束了，现在上神经网络，采用刚才的minmax归一化数据集，进行训练，模型结构如下（两步relu激活函数位于forward内）：

```
Model(
  (fc1): Linear(in_features=280, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=128, bias=True)
  (fc5): Linear(in_features=128, out_features=2, bias=True)
)
```

图 5: DNN模型结构

训练迭代200次，随机梯度下降，MultiStepLR学习率规划器，初始学习率为1e-3，在20与50的milestone处进行后移一位，batch大小32。

aaa	0.9868	0.99	0.99	0.99

图 6: DNN评测网站

原来DNN就能做到这个地步吗？

5.7 CNN

但我的模型都搭好了，不能不做啊，这里再使用CNN进行试验，模型参数如下，超参数与DNN相同。

```
CNN1D(
  (conv1): Conv1d(1, 16, kernel_size=(5,), stride=(1,))
  (pool1): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv1d(16, 32, kernel_size=(5,), stride=(1,))
  (pool2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv1d(32, 64, kernel_size=(5,), stride=(1,))
  (pool3): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=1344, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=2, bias=True)
)
```

图 7: CNN模型结构

毫无悬念的结果。

cnn	0.9868	0.99	0.99	0.99

图 8: CNN评测网站

5.8 RNN

这里再使用RNN进行试验，模型参数如下，超参数与DNN，CNN相同。

```
LstmRNN(
  (rnn): RNN(1, 64, num_layers=2, batch_first=True)
  (fc1): Linear(in_features=64, out_features=32, bias=True)
  (fc2): Linear(in_features=32, out_features=2, bias=True)
)
```

图 9: RNN模型结构

非常奇怪的现象出现了，XGB和LGB的全0全1在这里得到了解答，RNN意外地在同样超参数情况下没能收敛到最低值，结果就是acc稳定在78左右，最后预测的结果就是全0。

这样一来可以轻易的推论出一个粗浅的结论，那就是XGB和LGB的剪枝策略过于激进了，通过调节超参数树状集成学习方法也可以得到很好的效果，但是这很依赖超参数的设置。

6 实验结果与分析

表中数据汇总了所有评测网站中表现突出的模型，附上了主观的稳定性评价。

表 7: 实验结果

Metric	ACC	F1score	Recall	precision	stability
XGB RF	0.9934	0.9901	1	0.995	high
LGBM RF	0.9934	0.9901	1	0.995	high
RF	0.9868	0.99	0.99	0.99	high
GBDT	0.9868	0.99	0.99	0.99	medium
DNN	0.9868	0.99	0.99	0.99	low
CNN	0.9868	0.99	0.99	0.99	medium

7 开放性提出及思考

主观分析来看，正应了那句**“没有最好的模型，只有最适合的模型”**，所有模型都有其用武之地，随机森林算法在此次任务中的表现非常出色，起到决定性作用。

鉴于数据的表现过于逆天，完全可以采用**模板匹配**的方式进行测试，使用**波形叠加**的方法进行观测，找到所有最有可能有电动车的波形，打上1，这种方案在实验中**已经被证明可行**，效果极佳。

8 结论

通过使用基于随机森林算法的XGB和LGBM集成学习算法，都能获得极高的评分，并且模型的鲁棒性极强，精确度大概稳定在0.9934，是非常不错的表现。

参考文献

[HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[KGJV83] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[SKK18] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty, 2018.

[ZZW+23] Huan Ma. Qingyang Zhang, Changqing Zhang, Bingzhe Wu, Huazhu Fu, Joey Tianyi Zhou, and Qinghua Hu. Calibrating multimodal learning, 2023.