# 天池-广东工业智造大数据创新大赛

<Excerpt in index | 首页摘要>
**因为最近参加了天池的一个关于区分钢材瑕疵的多分类比赛，这里做个总结笔记**
**比赛主要的任务是训练一个高效的算法区分所给的几百张钢材中的瑕疵类型，简而言之就是一个多分类的比赛**
最后并没有取得很好的成绩，当做学习了，
Website：比赛
Github

<The rest of contents | 余下全文>

# 比赛思路

近期听了一个比赛分享，主题是关于时尚服装的属性预测。通常把它看作一个**多标签分类问题，预测其所有可能的属性标签。**觉得他的做比赛的思路和方案具有借鉴意义，这边写一下。

## 思路主要分为四个部分(以后比赛也可以从下面四个思路入手)：

### 1: Data argumentation

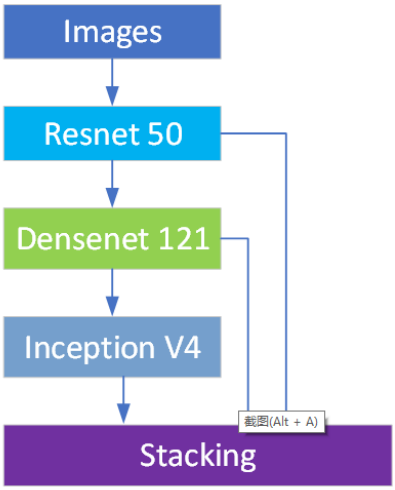
**Imbalance data distribution**

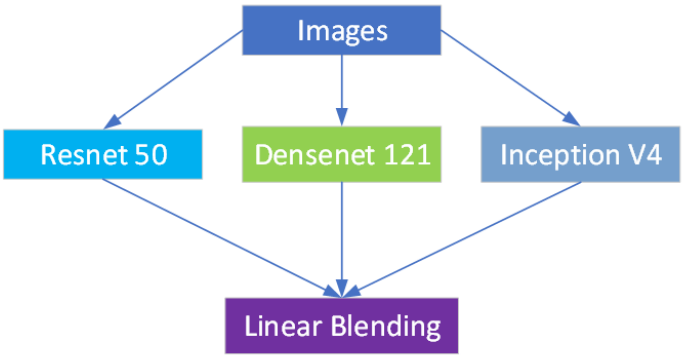
**Resampling the training set**

**训练集可以随便改，增加，重采样。但是验证集不能改变，尤其是正负样本的比例**

### 2: Transfer learning

Comparison of stacking and linear blending method.




### 3: Model fusion

### 4:The style relationship

# 比赛介绍


**数据源：**

大赛数据集里有1万份来自实际生产中有瑕疵的铝型材监测影像数据，每个影像包含一个或多种瑕疵。供机器学习的样图会明确标识影像中所包含的瑕疵类型。

**比赛规程：**

参赛者自行下载初赛数据，具体如下：

1. **参考学习数据量：**9月1日提供下载，300张图片，包含所有瑕疵的类型。用于参赛者设计图像识别算法和机器学习。
2. **初赛数据量：**3000张图片，包含所有瑕疵的类型。参赛者可以将自己算法识别的结果上传系统，识别率高的前100支团队晋级。

3. **复赛数据量：**5000张图片，包含所有瑕疵的类型。晋级复赛的参赛队伍在规定的时间内，通过算法自动识别照片中的瑕疵类型。综合计算识别张数、识别准确率、时长等因素计算出效率最高的6支队伍晋级决赛，参加在佛山南海举行的决赛答辩路演，产出最终获胜团队举行决赛颁奖。

**评价指标：**

$$平均准确率 = \frac{1}{N}\sum_{i=1}^{N}\frac{判定为i类并且正确图片个数}{判定为i类的图片数}$$

**结果：**


# 解决方案:

## Data Argumentation

### 数据集和验证集分割：

基于sklearn和keras的数据切分与交叉验证

### 数据均衡化：

什么是非均衡化数据

如何处理

1. 通过sampling技术，生成均衡的样本
2. 调整算法，允许算法改变权重weight变得cost-sensitive。
3. 采用集成学习思想， 将major类进行横向划分，形成小的均衡的样本集群

## Focal loss：

由于前正负样本不均衡的问题（感觉理解成简单-难分样本不均衡比较好）,这里提出一种**新的损失函数**，思路是希望那些hard examples对损失的贡献变大，使网络更倾向于从这些样本上学习。

```
gamma=2.
alpha=.25
def focal_loss_fixed(y_true, y_pred):
    pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
    pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))
    return -K.sum(alpha * K.pow(1. - pt_1, gamma) * K.log(pt_1))-K.sum((1
-alpha) * K.pow( pt_0, gamma) * K.log(1. - pt_0))
```

# 交叉验证（simple cross validation):

# Keras学习率调整:

## ReduceLROnPlateau

**当评价指标不在提升时，减少学习率。**当学习停滞时，减少2倍或10倍的学习率常常能获得较好的效果。该回调函数检测指标的情况，如果在patience个epoch中看不到模型性能提升，则减少学习率。

### 参数

- monitor：被监测的量
- factor：每次减少学习率的因子，学习率将以lr = lr*factor的形式被减少
- patience：当patience个epoch过去而模型性能不提升时，学习率减少的动作会被触发
- mode：‘auto’，‘min’，‘max’之一，在min模式下，如果检测值触发学习率减少。在max模式下，当检测值不再上升则触发学习率减少。
- epsilon：阈值，用来确定是否进入检测值的“平原区”
- cooldown：学习率减少后，会经过cooldown个epoch才重新进行正常操作
- min_lr：学习率的下限

## 代码

```
from keras.callbacks import ReduceLROnPlateau
reduce_lr = ReduceLROnPlateau(monitor='val_loss', patience=10, mode='aut
o')
model.fit(train_x, train_y, batch_size=32, epochs=5, validation_split=0.
1, callbacks=[reduce_lr])
```

## keras自定义评价函数

```
import tensorflow as tf
def metric_LB(y_true,y_pred):
    y_pred_label = K.argmax(y_pred,axis=-1)
    y_pred_ = K.one_hot(y_pred_label,num_classes=12)
    tp_fp = K.sum(y_pred_,axis=0)
    n = tf.count_nonzero(tp_fp,dtype='float32')
    tp = K.sum(y_true*y_pred_,axis=0)
    return K.sum(tp/(tp_fp+1e-6))/n
```

# 可视化CNN:

**relevant blog:**

Deep Visualization:可视化并理解CNN

tensorflow tutorials（九）：卷积神经网络可视化

凭什么相信你，我的CNN模型？（篇一：CAM和Grad-CAM)

Paper

在Learning Deep Features for Discriminative Localization这篇文章中，作者提出了CNN网络除了具有很强的图片处理，分类能力；**同时还能够针对图片中的关键部分进行定位，这个过程被称为Class Activation Mapping，简称CAM。**

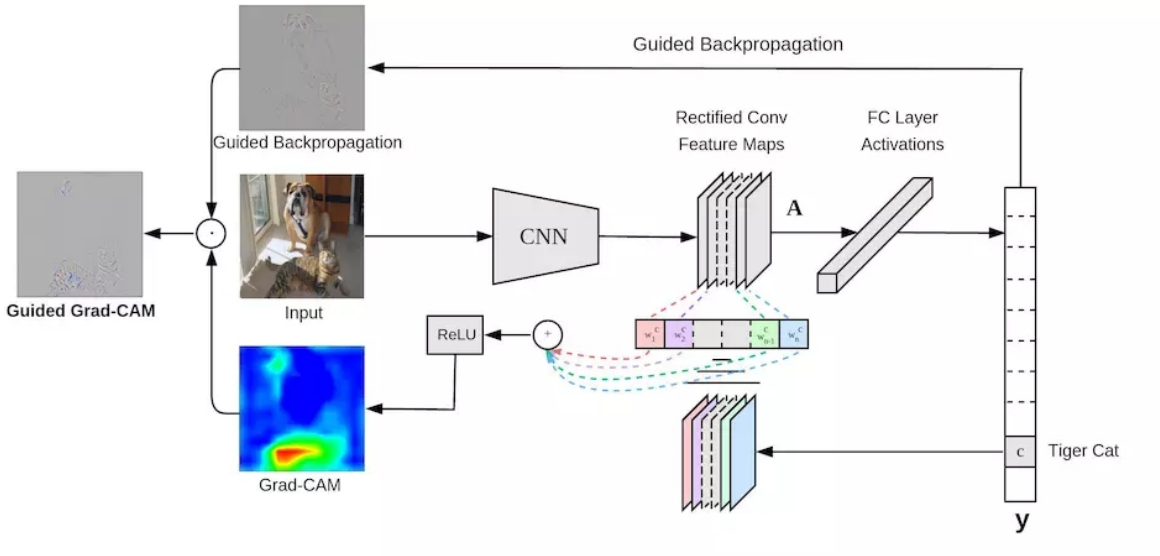
# Grad-CAM方法:

Grad-CAM的基本思路和CAM是一致的，也是通过得到每对特征图对应的权重，最后求一个加权和。但是它与CAM的主要区别在于求权重$w_k^c$的过程。

CAM通过替换全连接层为GAP层，重新训练得到权重，而Grad-CAM另辟蹊径，用梯度的全局平均来计算权重。事实上，经过严格的数学推导，Grad-CAM与CAM计算出来的权重是等价的。为了和CAM的权重做区分，定义Grad-CAM中第k个特征图对类别c的权重为$\alpha_k^c$，可通过下面的公式计算：

$$\alpha_k^c = \frac{1}{Z}\sum_i\sum_j\frac{\partial y^c}{\partial A_{ij}^k}$$

Grad-CAM的整体结构如下图所示：




# 代码实现：

## 管理GPU资源：

```
import os
os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"]="1"
```

## 导入数据并查看数据分布：

```
def load_data(path):
    print('[INFO] loading images...')
    data = []
    labels = []
    # grad the image paths and randomly shuffle them
    imagePaths = sorted(list(paths.list_images(path)))
#     print(imagePaths)
    random.seed(42)
    random.shuffle(imagePaths)
    # loop over the imput image
    for imagePath in imagePaths:
        image = cv2.imread(imagePath)
#         print(image)
        image = cv2.resize(image, (norm_size,norm_size))
        image = img_to_array(image)
        data.append(image)
```

```
        # extract the class label from the image path and update the labe
l list
        label = str(imagePath.split(os.path.sep)[-2])
#         print(label)
        labels.append(label)

    # scale the raw pixel intensities to the range(0,1)
    data = np.array(data,dtype="float") / 255.0
    labels = np.array(labels)

    labels = to_categorical(labels,num_classes=CLASS_NUM)
    return data,labels

def panda_data(label):
    number_list = label
    num_list = ['norm','defect1','defect2','defect3','defect4','defect
5','defect6','defect7','defect8','defect9','defect10','defect11']
    plt.figure(figsize=(20,10))
    plt.bar(range(len(number_list)), number_list, tick_label = num_list)
    plt.show()
```

## 训练数据:

```
# 当评价指标不在提升时，减少学习率
reduce_lr = ReduceLROnPlateau(monitor='val_metric_LB', patience=5,
mode='auto',factor=0.5,min_lr =0.5e-6)

def train(aug,trainX,trainY,testX,testY):
    # initialize the model
    print("[INFO] compiling model ...")

    model.compile(loss=focal_loss_fixed,
              optimizer=optimizers.Adam(lr=1e-4, decay=1e-6),
              metrics=['accuracy', metric_LB])
    #train the network
    print("[INFO] training network...")
    H = model.fit_generator(aug.flow(trainX, trainY, batch_size=BS),
        validation_data=(testX, testY), steps_per_epoch=len(trainX) // B
S,
        epochs=EPOCHS, verbose=1,callbacks=[checkpoint,reduce_lr],class_w
eight={0:1., 1:26.25806452, 2:6.832214, 3:21.42105263,

                                       4:11.30555556, 5:3.609929078,
6:13.12903226,

                                       7:18.85023256, 8:16.41917647,
9:29.07142857,
```

```
                                        10:4.8742515 ,  11:5.7197011234

                                          })

    # save the model to disk
    print("[INFO] serializing network...")

    # plot the training loss and accuracy
    plt.style.use("ggplot")
    plt.figure()
    N = EPOCHS
    plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
    plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
    plt.plot(np.arange(0, N), H.history["acc"], label="train_acc")
    plt.plot(np.arange(0, N), H.history["val_acc"], label="val_acc")
    plt.title("Training Loss and Accuracy on rolled_steel classifier")
    plt.xlabel("Epoch #")
    plt.ylabel("Loss/Accuracy")
    plt.legend(loc="lower left")
    plt.savefig("/home/weijia.wu/workspace/Kaggle/rolled_steel/Net_Gangec
ai/result/ResNet_weijiawu_20.png")
```

# 反馈与建议

- 微博：@柏林designer
- 邮箱：wwj123@zju.edu.cn