دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)
دانشکده مهندسی صنایع و سیستم‌های مدیریت

کنترل کیفیت پروانه‌ی پمپ شناور با Convolutional Neural Network

پروژه‌ی درس هوش مصنوعی

اعضای گروه:

حامد اعراب – ۹۹۲۵۰۰۳
فاطمه خدابنده – ۹۸۲۵۰۱۱
روشا مشتاقیان – ۹۹۲۵۰۸۰

استاد درس:

دکتر مرضیه زرین‌بال

پاییز و زمستان ۱۴۰۱

فهرست

مقدمه

سیستم‌های مکانیکی صنعتی به طور مداوم باهوش‌تر و پیچیده‌تر می‌شوند. بنابراین، نیاز روشنی به تحقیق و توسعه در مورد روش‌های داده محور و تکنیک‌های نظارت بر شرایط وجود دارد که باعث تشخیص سریع و قابل اعتماد به صورت خودکار می‌شوند.

ماشین‌های چرخان به طور گسترده در صنعت تولید استفاده می‌شوند که معمولاً برای مدت طولانی در شرایط سخت کار می‌کنند. خرابی‌های ناگهانی که بر روی اجزای اصلی ماشین مانند چرخ‌دنده‌ها و یاتاقان‌ها رخ می دهد ممکن است منجر به شکست غیرمنتظره دستگاه‌ها شود و باعث ضربه اقتصادی، آلودگی محیط زیست و تلفات انسانی شود. پس تشخیص زودرس نقص و خرابی چنین مؤلفه‌های ماشین‌آلات در حال چرخش آن‌ها بسیار مهم است و از تصادفات فاجعه بار در کاربردهای صنعتی جلوگیری می‌کند.

CNN

در یادگیری عمیق، شبکه عصبی کانولوشنال (CNN یا ConvNet) یک کلاس از شبکه‌های عصبی عمیق است که معمولاً برای تجزیه‌وتحلیل تصاویر بصری استفاده می‌شود و تصویر ورودی را دریافت می‌کند و به هر یک از اشیا/جنبه‌های موجود در تصویر میزان اهمیت (وزن‌های قابل یادگیری و بایاس) تخصیص می‌دهد و قادر به متمایزسازی آن‌ها از یکدیگر است.آنها همچنین بر اساس معماری وزن‌های مشترک و ویژگی‌های تغییرناپذیری ترجمه آن‌ها، به عنوان شبکه‌های عصبی مصنوعی تغییرناپذیر یا تغییرناپذیر فضایی شناخته می‌شوند. آن‌ها در تشخیص تصویر و ویدئو، سیستم‌های توصیه‌کننده، طبقه‌بندی تصاویر، تجزیه‌وتحلیل تصاویر پزشکی، پردازش زبان طبیعی، رابط‌های مغز و کامپیوتر و سری زمانی مالی کاربرد دارند.

دیتاست

دیتاست این پروژه از سایت Kaggle دریافت شد. تصاویر دیتاست از نمای بالای پروانه پمپ شناور است. این مجموعه داده شامل ۷۳۴۸ داده تصویری تقویت‌شده (Augmented) با اندازه (۳۰۰*۳۰۰) پیکسل که دارای مقیاس (خاکستری) سیاه و سفید می‌باشد. همچنین مجموعه داده شامل دو مجموعه تصاویر معیوب و غیرمعیوب بوده که دارای دو پوشه آموزشی و آزمایشی می‌باشند. هر دو پوشه آموزش و آزمایش شامل تصاویر معیوب و غیرمعیوب‌اند. مجموعه آموزش شامل ۳۷۵۸ تصویر غیرمعیوب و ۲۸۷۵ تصویر معیوب و مجموعه آزمایش شامل ۴۵۳ تصویر غیرمعیوب و ۲۶۲ تصویر معیوب است.

توضیحات درمورد پروژه

این پروژه اساساً یک مساله طبقه‌بندی است. از آنجایی که با تصاویر سروکار داریم، به دلیل توانایی‌هایی که در پردازش تصویر دارد، تصمیم گرفته ایم آن را با یک شبکه عصبی کانولوشن حل کنیم. مراحل پروژه به صورت زیر است:

۱) دریافت داده‌ها
۲) پیش‌پردازش داده‌ها
۳) تعریف مدل شبکه‌ی عصبی
۴) آموزش و اعتبارسنجی شبکه‌ی عصبی

تصاویر پروانه پمپ معیوب و غیرمعیوب را در پوشه‌های مربوطه بارگذاری می‌کنیم. توجه داشته باشید که داده‌های ما به‌طور پیش‌فرض به بخش‌های آموزش و اعتبارسنجی تقسیم شدند.

پیش‌نیازها

۱) Tensorflow، برای تعریف، آموزش و اعتبارسنجی مدل شبکه‌ی عصبی
۲) Numpy، برای کار با آرایه‌ها
۳) Matplotlib، برای به نمایش درآوردن تصاویر
۴) Visualkeras، برای مصورسازی مدل شبکه‌ی عصبی

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import visualkeras as vk
```

مصورسازی داده‌ها

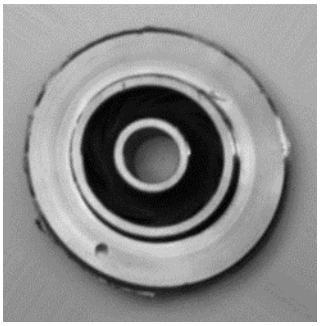
یک تابع برای بارگیری تصاویر در حالت مقیاس خاکستری تعریف می کنیم. سپس، تابع دیگری را برای دریافت تصویر و تجسم آن با استفاده از Matplotlib تعریف می کنیم. این تابع تصویر را به یک آرایه تبدیل می کند و مقیاس مقادیر آن را به [0, 1] تغییر می‌دهد.

```
def get_image(path):
    return tf.keras.preprocessing.image.load_img(path, color_mode = 'grayscale')

def visualize_image(image):
    plt.figure()

    plt.imshow(
        tf.keras.preprocessing.image.img_to_array(image) / 255,
        cmap = 'gray',
    )
```

```
    plt.show()
```

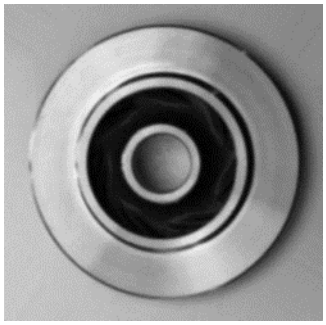
نمونه پروانه پمپ معیوب

```
defective_image_example = get_image('./data/test/defective/cast_def_0_108.jpeg')

visualize_image(defective_image_example)
```



نمونه پروانه پمپ غیرمعیوب

```
non_defective_image_example = get_image(
    './data/test/non_defective/cast_ok_0_1026.jpeg'
)

visualize_image(non_defective_image_example)
```



پیش‌پردازش داده‌ها

اکنون باید داده‌ها را با بارگیری و تغییر مقیاس، پیش‌پردازش کنیم. برای بهبود توانایی‌های پردازشی مدل خود، به‌طور تصادفی داده‌های آزمایشی را بزرگ‌نمایی و تغییر می‌دهیم تا دیدن اشیا از زوایای مختلف و فواصل مختلف را شبیه‌سازی کنیم.

```
train_data = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale = 1 / 255,
    zoom_range = 0.2,
    shear_range = 0.2,
).flow_from_directory(
```

```
    './data/train',
    class_mode = 'binary',
    batch_size = 8,
    target_size = (64, 64),
    color_mode = 'grayscale',
)

test_data = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale = 1 / 255,
).flow_from_directory(
    './data/test',
    class_mode = 'binary',
    batch_size = 8,
    target_size = (64, 64),
    color_mode = 'grayscale',
)
```

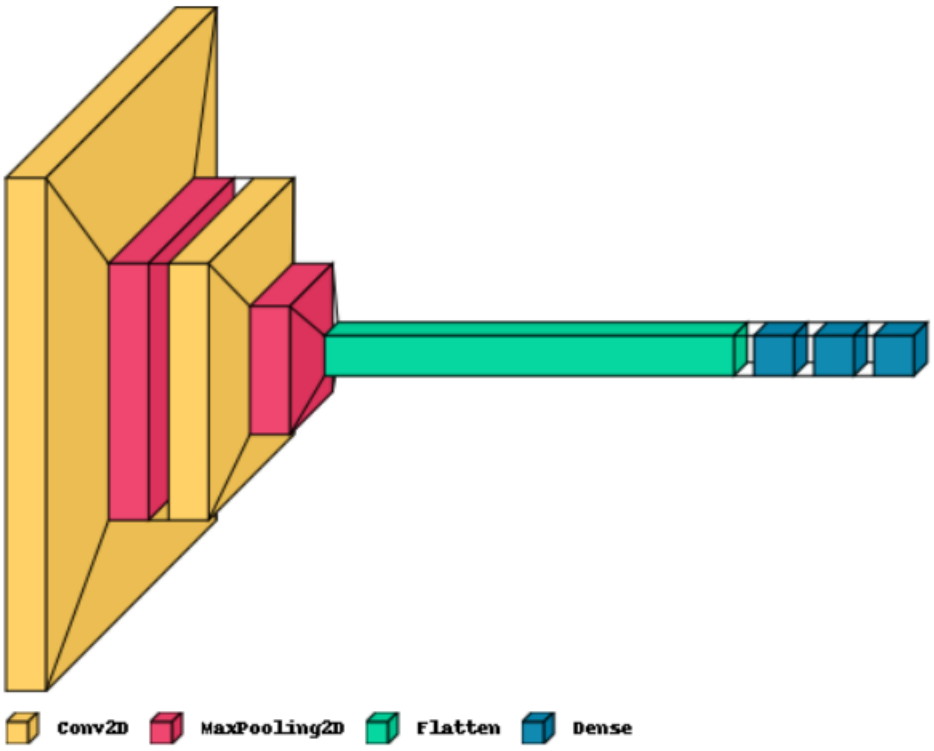
```
Found 6633 images belonging to 2 classes.
Found 715 images belonging to 2 classes.
```

تعریف مدل CNN

مدل CNN ما دو سری لایه‌ی Convolution و Max-Pooling خواهد داشت. سپس خروجی را مسطح کرده و به یک MLP با دو لایه‌ی مخفی منتقل می‌کنیم. در نهایت مدل خود را با بهینه‌ساز Adam و تابع Binary Cross-Entropy کامپایل می‌کنیم.

```
model = tf.keras.models.Sequential()

model.add(
    tf.keras.layers.Conv2D(
        filters = 8,
        kernel_size = 3,
        activation = 'relu',
        padding = 'same',
        input_shape = (64, 64, 1),
    )
)

model.add(tf.keras.layers.MaxPooling2D(pool_size = 2))

model.add(
    tf.keras.layers.Conv2D(
        filters = 8,
        kernel_size = 3,
```

```
        activation = 'relu',
        padding = 'same',
    )
)

model.add(tf.keras.layers.MaxPooling2D(pool_size = 2))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(units = 16, activation = 'tanh'))
model.add(tf.keras.layers.Dense(units = 16, activation = 'relu'))

model.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))

model.compile(
    optimizer = 'adam',
    loss = 'binary_crossentropy',
    metrics = ['binary_accuracy'],
)

vk.layered_view(model, legend = True)
```



آموزش مدل CNN

مدل را در ۲۰ اپوک آموزش می‌دهیم و بهترین آن را در یک فایل ذخیره می‌کنیم.

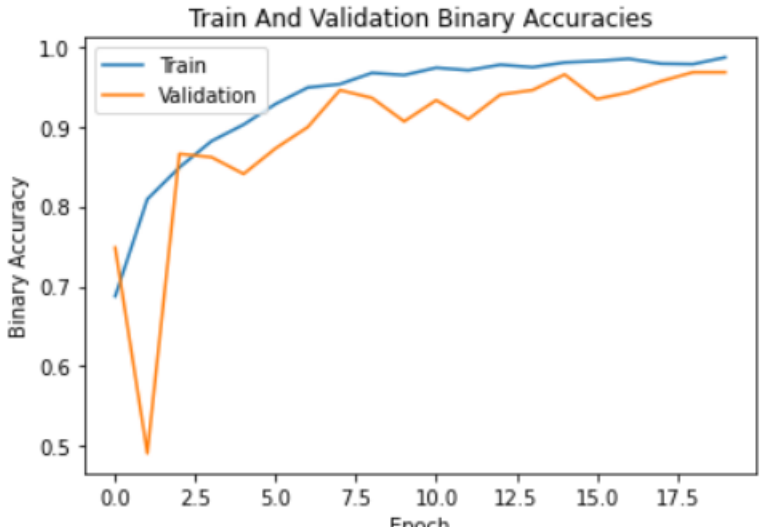
```
history = model.fit(train_data, validation_data = test_data, epochs = 20)

model.save('classifier_model.h5')
```

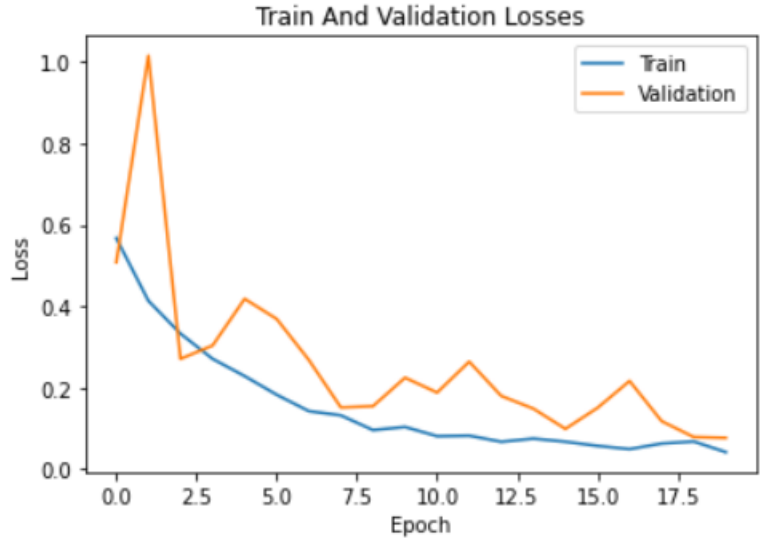
```
...

Epoch 19/20 830/830 [==============================] - 14s 17ms/step - loss: 0.0675
- binary_accuracy: 0.9780 - val_loss: 0.0785 - val_binary_accuracy: 0.9678

Epoch 20/20 830/830 [==============================] - 14s 17ms/step - loss: 0.0414
- binary_accuracy: 0.9864 - val_loss: 0.0764 - val_binary_accuracy: 0.9678
```

نمودار دقت و زیان مدل CNN

در نهایت دقت و زیان پیش‌بینی مدل را در تمام اپوک‌ها رسم می‌کنیم.

```
plt.figure()

plt.title('Train And Validation Binary Accuracies')

plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])

plt.xlabel('Epoch')
plt.ylabel('Binary Accuracy')

plt.legend(['Train', 'Validation'], loc = 'upper left')

plt.figure()

plt.title('Train And Validation Losses')

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.xlabel('Epoch')
plt.ylabel('Loss')

plt.legend(['Train', 'Validation'], loc = 'upper right')
```





آزمایش مدل CNN

در مرحله بعدی مدل خود را با نمونه تصـاویر پروانه پمپ معیوب و غیرمعیوب آزمایش می‌کنیم. ابتدا یک تابع «پیش‌بینی» تعریف می‌کنیم تا کارها آسان‌تر شود.

```
def predict(image):
    result = model.predict(
        np.expand_dims(
            tf.keras.preprocessing.image.img_to_array(
                image.resize((64, 64)),
            ) / 255,
            axis = 0
        )
    )
```

```
    if result[0][0] <= 0.5:
        print(f'Defective (Prediction Value: {result[0][0]})')
    else:
        print(f'Non-Defective (Prediction Value: {result[0][0]})')
```

آزمایش با نمونه پروانه پمپ معیوب

```
predict(defective_image_example)
```

```
1/1 [==============================] - 0s 97ms/step
Defective (Prediction Value: 9.853595202002907e-07)
```

آزمایش با نمونه پروانه پمپ غیرمعیوب

```
predict(non_defective_image_example)
```

```
1/1 [==============================] - 0s 18ms/step
Non-Defective (Prediction Value: 0.9985570311546326)
```

نتیجه‌گیری

مدل ما می‌تواند با دقت تقریبی ۹۶٪ معیوب یا غیرمعیوب بودن پروانه‌های پمپ شـــناور را پیش‌بینی کند.