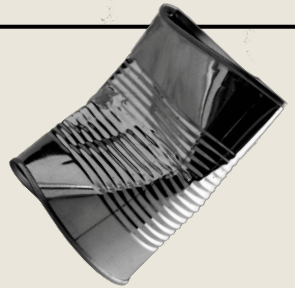
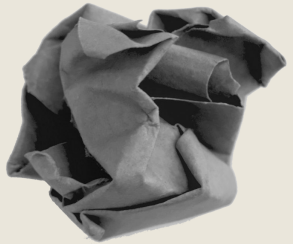
Recyclable Waste Classification

Kaan Kazancoglu,
Hector He,
Yu Ting Hung

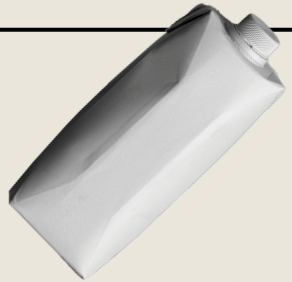
TABLE OF CONTENTS





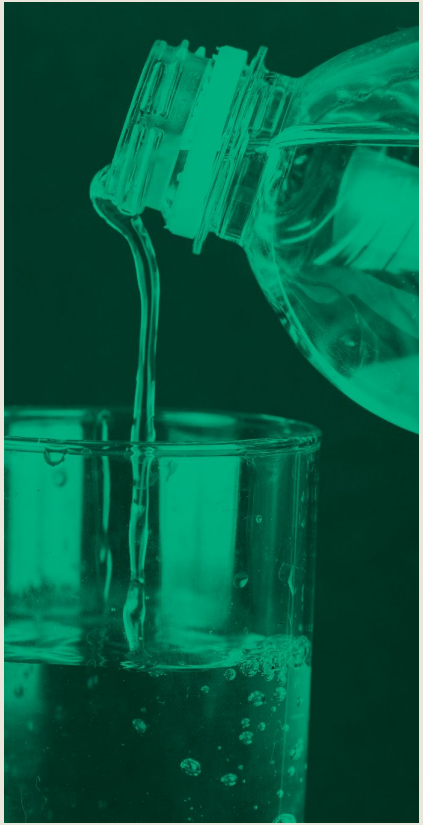
“Most Americans want to recycle, as they believe recycling provides an opportunity for them to be responsible caretakers of the Earth. However, it can be difficult for consumers to understand what materials can be recycled”

—U.S. Environmental Protection Agency



Motivations

Improve recycling efficiency, reduce labor cost, reduce environmental pollution, enhance public health, and therefore build stronger communities.



Goal

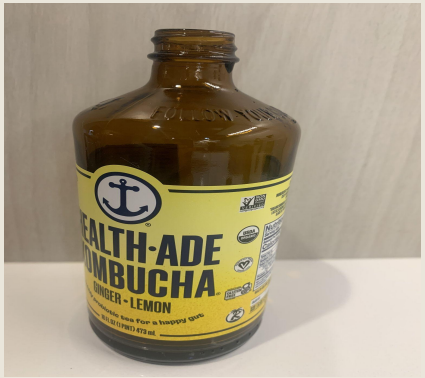
Label: waste category (plastic, paper, metal, glass)

Input: a colored image of the waste item

DATA SOURCE:
WE TOOK OUR OWN PHOTOS!

Paper x30

cereal boxes,
newspapers,
cardboards...



Glass x30

wine bottles,
pickle jars,
glass vases...

Plastics x30

shampoo containers,
water bottles...

Metal x30

food cans,
beverage cans!

Data Cleaning

- Convert the photos from .heic to .jpeg format
- Resize the images to (224,224,3)

Training process

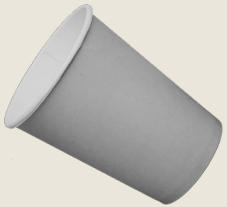
Baseline Model

Upgraded
We improve our own NN model

- v1.0
  Data Augmentation + Weight Regularization/Dropout + Additional Nodes/Layers
- v2.0
  Data Augmentation + Residual Connection + Regularization/Dropouts

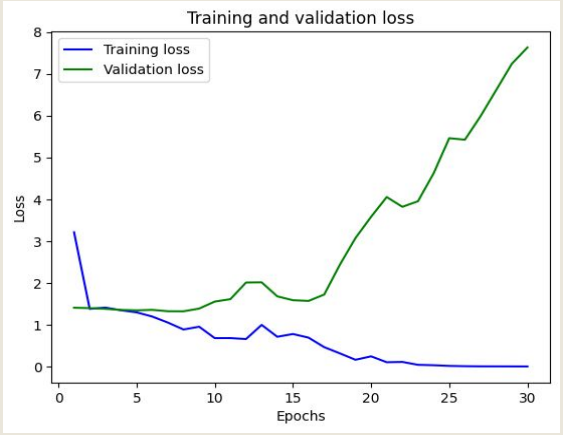
Pre-trained
We stand upon the shoulders of VVG16

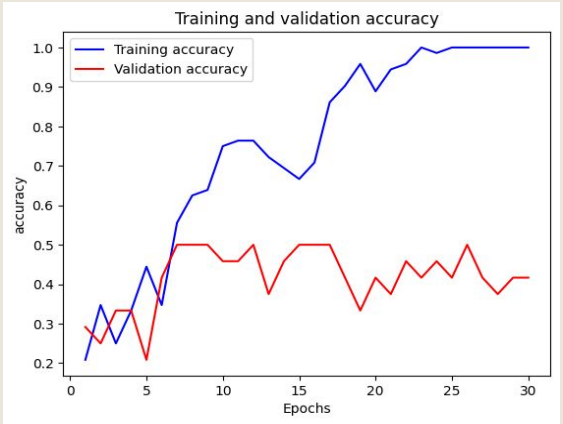
- Data Augment
- K-fold v1.0 - v2.5



Baseline Model

```python
def create_model():
    inputs = Input(shape=(224, 224, 3))

    x = Conv2D(32, kernel_size=3, activation='relu', padding='same')(inputs)
    x = MaxPooling2D(pool_size=2)(x)
    x = Conv2D(64, kernel_size=3, activation='relu', padding='same')(x)
    x = MaxPooling2D(pool_size=2)(x)
    x = Conv2D(128, kernel_size=3, activation='relu', padding='same')(x)
    x = MaxPooling2D(pool_size=2) (x)
    x = Conv2D(256, kernel_size=3, activation="relu", padding ="same")(x)
    x = Flatten()(x)
    outputs = Dense(4, activation='softmax')(x)
    model = Model(inputs=inputs, outputs=outputs)

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    return model
```





What are the problems?

1. the model starts overfitting very soon, right after ~ epoch 10
2. the best validation accuracy achieved is only around 0.5, far from desired
3. there are way too many parameters

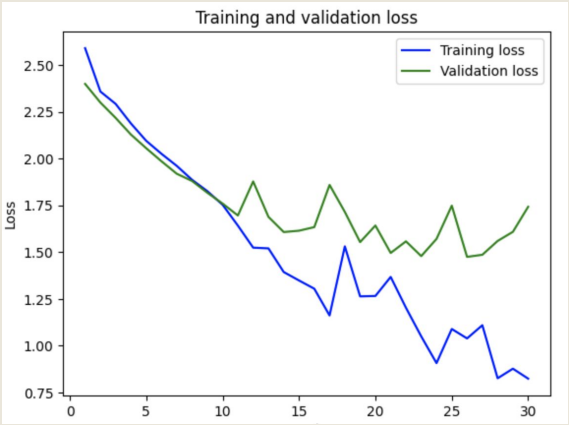
Upgrade Model 1.0

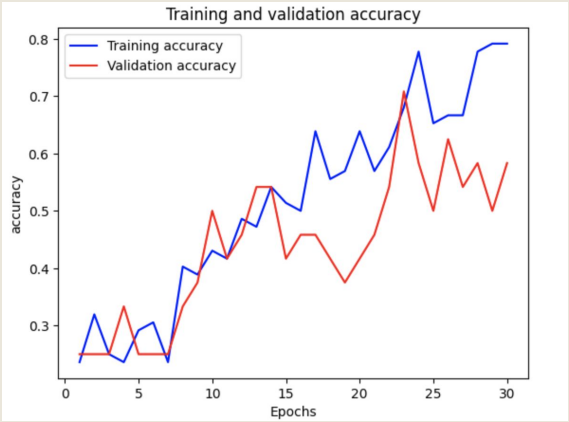
Data Augmentation +
Weight Regularization/Dropout +
Additional Nodes/Layers

- first used data augmentation to avoid overfitting due to a small training set
- we employed both L2 regularization at each Conv2D layer along with a dropout term at the end of the baseline model right before the dense layer
- we added another MaxPooling2D layer and another Conv2D layer at the end of the baseline model

What are the problems?

1. the model starts overfitting very soon, right after ~ epoch 10
2. the best validation accuracy achieved is only around 0.5, far from desired
3. there are way too many parameters

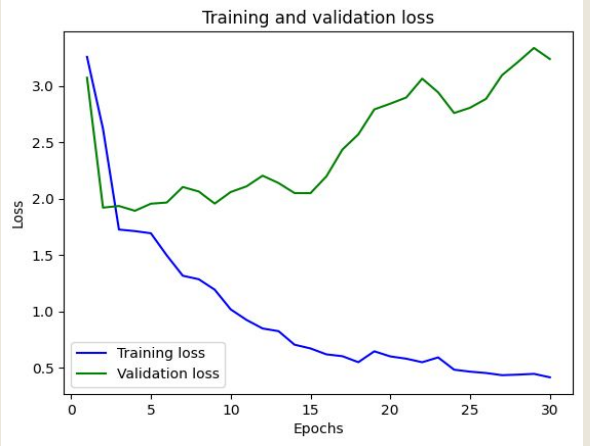


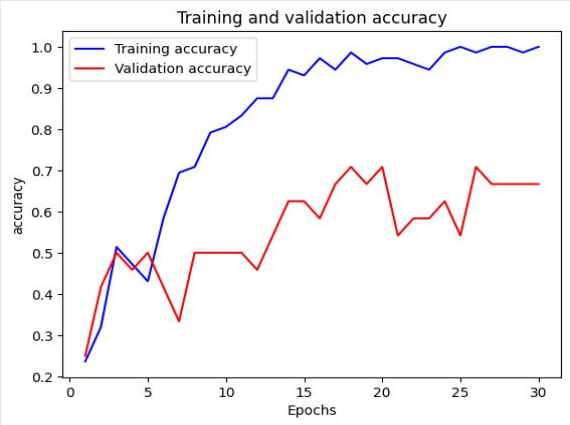


Upgrade Model 2.0

Data Augmentation + Residual Connection
+ Regularization/Dropouts

- for the obvious reason we kept the data augmentation as well as the regularization/dropout terms
- we applied residual connection here since our model has become too deep



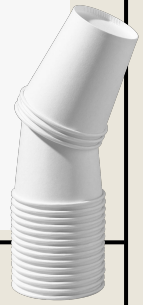


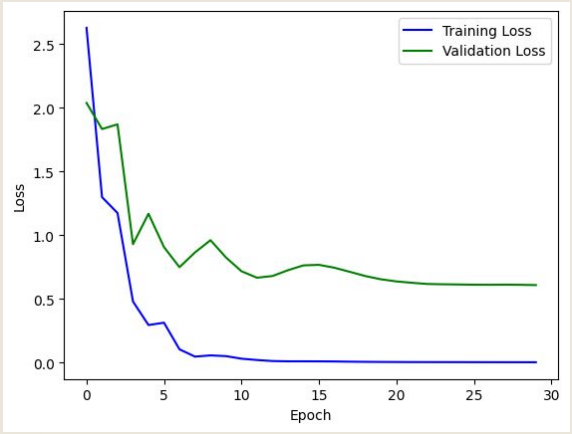
Pre-trained Model: data augmentation

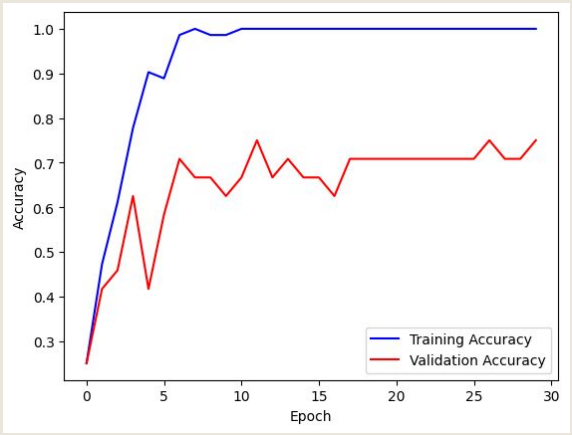
```
data_augmentation = keras.Sequential(
                [keras.layers.RandomFlip("horizontal"),
                 keras.layers.RandomRotation(0.5),
                 keras.layers.RandomZoom(0.2)])
```

GlobalAveragePooling is better for small dataset with fewer trainable parameters, and usually used for final pooling layer

```
# Load the pre-trained model
pretrained_model = keras.applications.vgg16.VGG16(weights="imagenet",
                                                   include_top = False, input_tensor=data_augmentation(inputs))

# Freeze the weights of the pre-trained layers
pretrained_model.trainable = False

# Customize the top layer
cuAu_model = tf.keras.Sequential()
cuAu_model.add(pretrained_model)
cuAu_model.add(tf.keras.layers.GlobalAveragePooling2D())
cuAu_model.add(tf.keras.layers.Dense(4, activation='softmax'))
```







What are the problems?

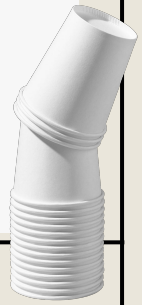
1. the model overfits the training dataset very soon
2. the best validation accuracy achieved is around 0.75

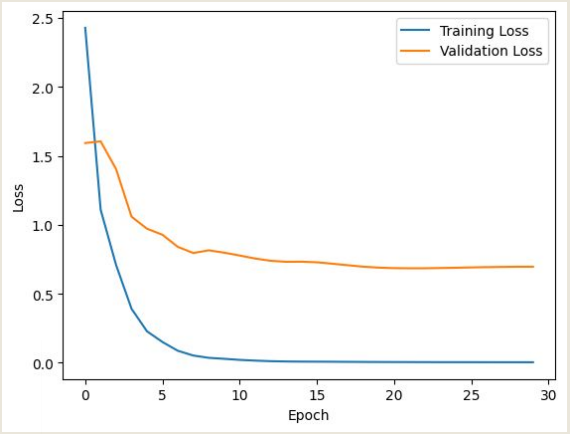
Pre-trained Model:

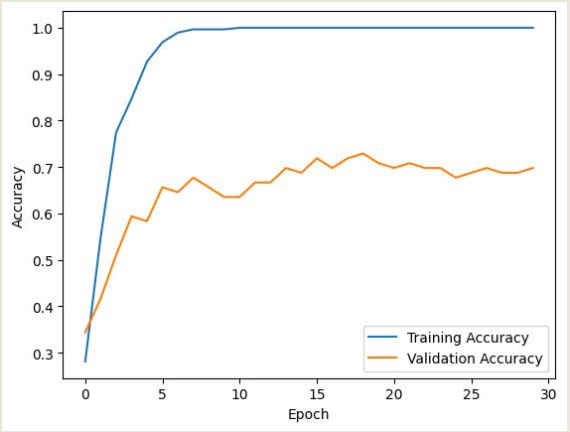
K-fold v1.0

Shuffle = False
K = 4

```
def pretrained_model():
    inputs = Input(shape=(224, 224, 3))
    pretrained_model = keras.applications.vgg16.VGG16(weights="imagenet",
                                                      include_top = False, input_shape = (224, 224, 3))

    # Freeze the weights of the pre-trained layers
    pretrained_model.trainable = False

    x =  Flatten()(pretrained_model.output)
    outputs = Dense(4, activation='softmax')(x)

    model = Model(inputs=pretrained_model.inputs, outputs=outputs)
    model.compile( loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

    return model
```







What are the problems?

1. the model starts overfitting from epoch 7 and reaching 1.0 in training accuracy
2. the best validation accuracy achieved is over 0.79

Pre-trained Model:

K-fold v2.0

Shuffle = True
K = 4
Epoch = 30

```
Epoch 3/30
3/3 [==============================] - 80s 25s/step - loss: 0.6943 - accuracy: 0.7083 - val_loss: 0.9464 - val_accuracy: 0.5000
Epoch 4/30
3/3 [==============================] - 79s 26s/step - loss: 0.4090 - accuracy: 0.8889 - val_loss: 1.1132 - val_accuracy: 0.4583
Epoch 5/30
3/3 [==============================] - 85s 27s/step - loss: 0.2736 - accuracy: 0.9583 - val_loss: 0.8038 - val_accuracy: 0.6250
Epoch 6/30
3/3 [==============================] - 78s 23s/step - loss: 0.1543 - accuracy: 0.9444 - val_loss: 0.6504 - val_accuracy: 0.7083
Epoch 7/30
3/3 [==============================] - 82s 25s/step - loss: 0.0880 - accuracy: 1.0000 - val_loss: 0.5461 - val_accuracy: 0.8750
Epoch 8/30
3/3 [==============================] - 78s 24s/step - loss: 0.0684 - accuracy: 1.0000 - val_loss: 0.5535 - val_accuracy: 0.8333
Epoch 9/30
3/3 [==============================] - 80s 27s/step - loss: 0.0497 - accuracy: 1.0000 - val_loss: 0.5525 - val_accuracy: 0.7500
Epoch 10/30
3/3 [==============================] - 78s 35s/step - loss: 0.0343 - accuracy: 1.0000 - val_loss: 0.5849 - val_accuracy: 0.7083
Epoch 11/30
3/3 [==============================] - 80s 36s/step - loss: 0.0240 - accuracy: 1.0000 - val_loss: 0.6221 - val_accuracy: 0.6667
Epoch 12/30
3/3 [==============================] - 83s 26s/step - loss: 0.0211 - accuracy: 1.0000 - val_loss: 0.6121 - val_accuracy: 0.7083
Epoch 13/30
3/3 [==============================] - 79s 24s/step - loss: 0.0176 - accuracy: 1.0000 - val_loss: 0.5852 - val_accuracy: 0.8333
Epoch 14/30
3/3 [==============================] - 79s 25s/step - loss: 0.0138 - accuracy: 1.0000 - val_loss: 0.5679 - val_accuracy: 0.8333
Epoch 15/30
3/3 [==============================] - 99s 36s/step - loss: 0.0121 - accuracy: 1.0000 - val_loss: 0.5586 - val_accuracy: 0.8333
```

In general, if the dataset is small, it is recommended to use `shuffle=True` to ensure that the model is trained on a more diverse set of data. However, if there is some inherent structure in the data, you may want to set `shuffle=False` to preserve this structure in the folds. Additionally, you can experiment with both settings and evaluate the model's performance to determine which setting works best for your specific use case.

What are the problems?

1. the model starts overfitting after epoch 7 and stops improving after epoch 10
2. the best validation accuracy achieved is over 0.83
3. should reduce the # of epochs or implement dropout / learning rate, but we only use the pre-trained model

Pre-trained Model:

K-fold v2.5

Shuffle = True
Learning rate = 0.001
Epoch = 15

```
Epoch 1/10
3/3 [==============================] - 82s 26s/step - loss: 2.5452 - accuracy: 0.3000 - val_loss: 1.9350 - val_accuracy: 0.2917
Epoch 2/10
3/3 [==============================] - 81s 38s/step - loss: 1.0590 - accuracy: 0.6429 - val_loss: 2.3719 - val_accuracy: 0.4167
Epoch 3/10
3/3 [==============================] - 76s 22s/step - loss: 1.2487 - accuracy: 0.5571 - val_loss: 1.4693 - val_accuracy: 0.5000
Epoch 4/10
3/3 [==============================] - 79s 23s/step - loss: 0.4087 - accuracy: 0.9143 - val_loss: 1.1776 - val_accuracy: 0.4583
Epoch 5/10
3/3 [==============================] - 78s 35s/step - loss: 0.2620 - accuracy: 0.9286 - val_loss: 1.2364 - val_accuracy: 0.5000
Epoch 6/10
3/3 [==============================] - 78s 25s/step - loss: 0.2539 - accuracy: 0.8857 - val_loss: 1.0623 - val_accuracy: 0.4167
Epoch 7/10
3/3 [==============================] - 98s 35s/step - loss: 0.1077 - accuracy: 0.9714 - val_loss: 0.9062 - val_accuracy: 0.5417
Epoch 8/10
3/3 [==============================] - 76s 22s/step - loss: 0.0552 - accuracy: 0.9857 - val_loss: 1.0629 - val_accuracy: 0.5000
Epoch 9/10
3/3 [==============================] - 80s 24s/step - loss: 0.0644 - accuracy: 0.9857 - val_loss: 1.0892 - val_accuracy: 0.5417
Epoch 10/10
3/3 [==============================] - 77s 22s/step - loss: 0.0534 - accuracy: 0.9857 - val_loss: 0.9911 - val_accuracy: 0.6250
```

What are the problems?

1. the # of epochs isn't sufficient for the model to learn well
2. the best validation accuracy achieved is 0.65, not high as we expected
3. DO NOT CHANGE THE DATA IN DRIVE WHILE RUNNING MODEL

"FileNotFoundError: [Errno 2] No such file or directory: 'drive/My Drive/xxxxxx/images_heic_to_jpeg/metal_heic_to_jpeg/ezgif-1-064ea82f3f.jpg'"

Results

```
✓ 25s
1 model.evaluate(test_generator,verbose=1)

1/1 [==============================] - 20s 20s/step - loss: 0.6492 - accuracy: 0.7308
[0.6491619944572449, 0.7307692170143127]
```

Solid waste management

RESOURCES

- ChatGPT
- https://www.epa.gov/circulareconomy/us-recycling-system#CurrentChallenges
- https://forums.fast.ai/t/shuffle-true-or-false/1438/2
- https://github.com/AshishBarvaliya/Face-detection-using-vgg16/blob/master/face_detection.ipynb