Teaching Your Computer to See: Using Computer Vision to Detect Defects

Presentation # AL-046

Scott Koval is a Sr. Data Scientist with Pinnacle Solutions, Inc. He has a Bachelor of Science in Psychology from Virginia Tech, a Master of Arts in Experimental Psychology from Radford University, and a Master of Science in Applied Statistics from Kennesaw State University. In addition to academic degrees, he also possesses five SAS certifications.

Scott takes pleasure in working on all parts of the data life cycle and is most interested in Big Data, IoT, Statistics/Analytics, and Natural Language Processing.

Teaching Your Computer to See: Using Computer Vision to Detect Defects

Presentation # AL-046

Scott Koval
Pinnacle Solutions, Inc
Indianapolis, IN

Outline

- Computer Vision (CV) Review
- Use Cases of CV
- Kaggle
  - Casting Product Image Data for Quality Inspection
  - Examples of Casting Non-defects and Defects
- SAS Viya and Deep Learning
  - Overview of Code and Results
- Wrap-up

Computer Vision

- What is Computer Vision?
  - Earliest experiments were in 1950s, while commercial use started in 1970s.
  - Type of Artificial Intelligence that is trained off digital images.
    - Deep Learning models that are created by allowing the computer to recognize patterns and adjust weights and biases in a neural network.
    - Examples of different methods: Convolutional Neural Network (CNN), You Only Look Once (YOLO), Generative Adversarial Network (GAN)
  - Uses include, but are not limited to:
    - Classification / Image Recognition
    - Recognition / Object Detection
    - Segmentation / Edge Detection
    - Estimation of Keypoints

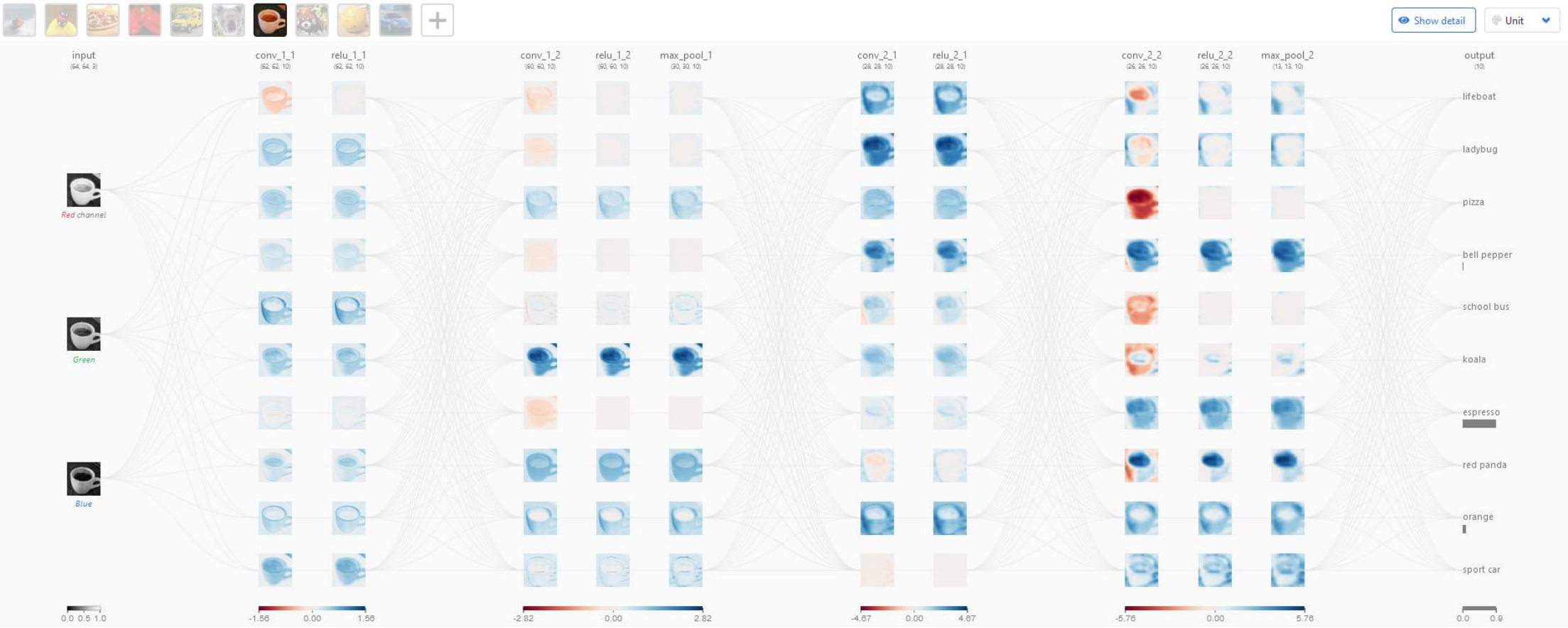
https://www.sas.com/en_us/insights/analytics/computer-vision.html

Classification

- Task is to classify or recognize what the image is.
- Popular method is using a CNN
  - Input -> Layers -> Output
  - Training adjusts weights and biases between neuron of each layer.
  - Image is classified by which output class has the highest score.
- This is the type of CV used in this presentation.

https://raw.githubusercontent.com/dusty-nv/jetson-inference/master/docs/images/imagenet-orange.jpg

CNN Example Diagram

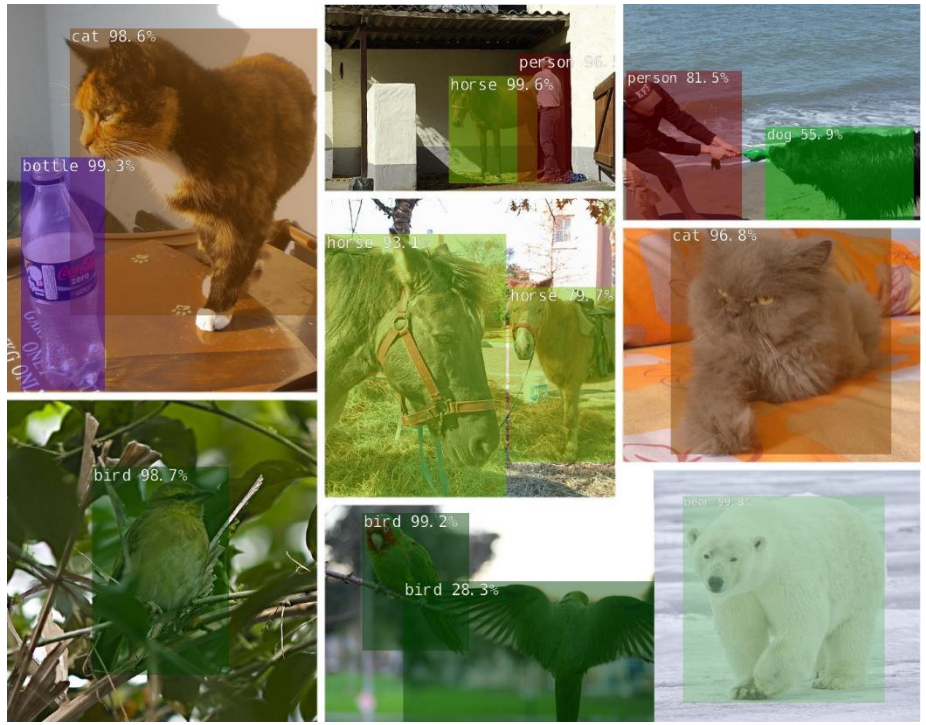


#MWSUG2024 #AL46

https://poloclub.github.io/cnn-explainer/

Object Detection

- Model trained to identify specific objects within an image.
- You Only Look Once (YOLO)
  - Breaks image up into grid and applies a classification probability.
  - Bounding boxes are formed around areas of high confidence of each class detected.



https://raw.githubusercontent.com/dusty-nv/jetson-inference/master/docs/images/detectnet.jpg

https://arxiv.org/pdf/1506.02640

Segmentation

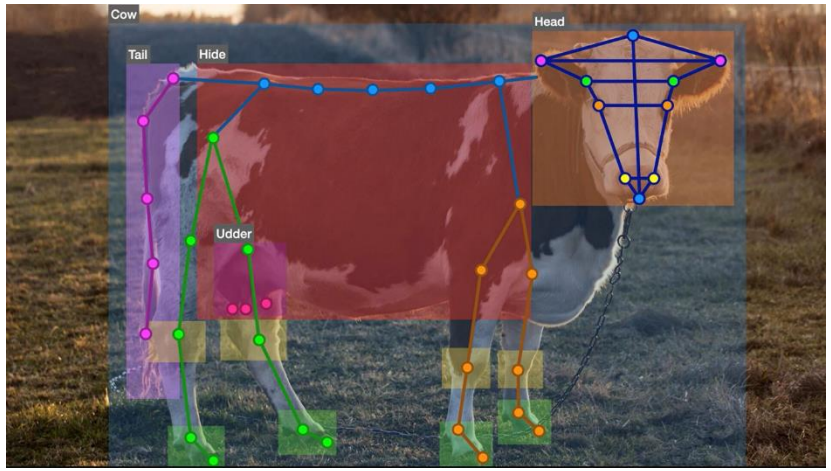
- Type of Classification, but at the pixel level.
- Fully Convolutional Network (FCN)
- Useful for understanding surrounding environment.
  - Road, cars, people, signs, etc…
  - Autonomous vehicles/drones

https://raw.githubusercontent.com/dusty-nv/jetson-inference/master/docs/images/segmentation.jpg

https://arxiv.org/abs/1605.06211

Estimation of Keypoints

- Attempts to locate various keypoints within an image.
- poseNet based off a CNN model.
- Not limited to just humans.



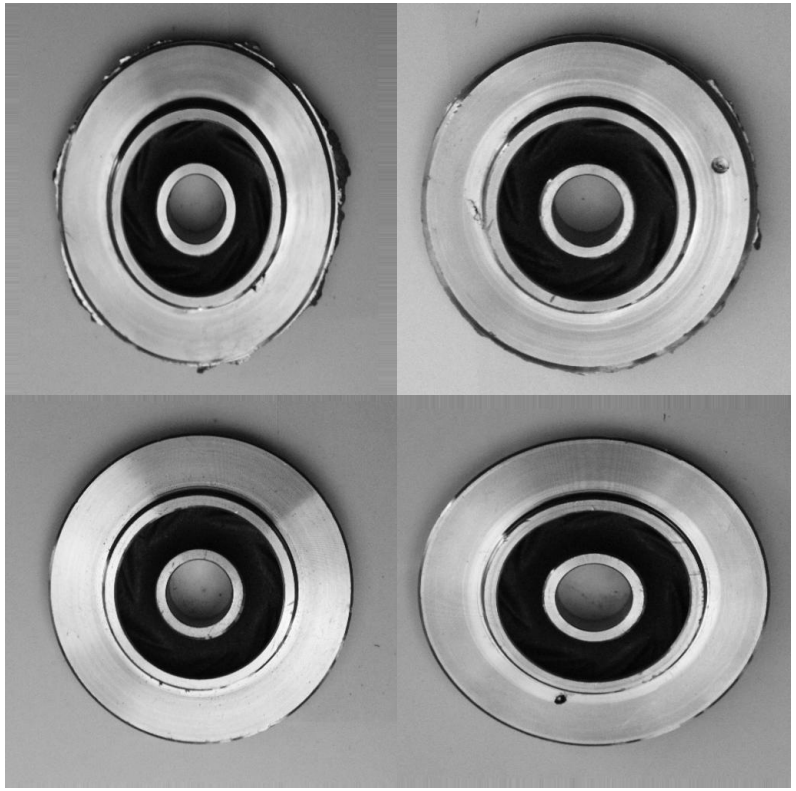
Building BETSY, World's First AI Ranch Hand

Kaggle

- Website focused on data science and machine learning.
  - Hosts datasets and allows users of all skill levels to learn, collaborate, and compete.
  - Wide variety of data across many industries.
- This presentation used manufacturing images
  - Casting Product Image Data for Quality Inspection
  - Collection of 7,348 gray-scaled images of both non-defective and defective castings.
  - Task was to build a model that would classify images.

Casting Product Image Data

Non-Defective



Defective

SAS Viya Deep Learning

- Modeling was performed using SAS Viya 3.5 environment.
- DLPy Python library for SAS Viya Deep Learning API
  - https://github.com/sassoftware/python-dlpy
  - Trained a CNN model for classification
- Other Python packages included
  - OS
  - SWAT
  - Matplotlib

SAS Viya Deep Learning

```
[1]
import os
os.environ['CAS_CLIENT_SSL_CA_LIST'] = r"/data/vault-ca.crt"

[2]
import swat
conn = swat.CAS(hostname='                                  ', port=5570, username=              , password=              )

[3]
display(conn)

CAS('                                  ;', 5570, '              ', protocol='cas', name='py-session-1', session='f725c275-d079-e64a-8843-5b2e283a5b26')

[4]
type(conn)

swat.cas.connection.CAS

[5]
import dlpy
from dlpy.splitting import two_way_split
from dlpy.applications import *
import matplotlib.pyplot as plt
%matplotlib inline

[6]
print(dlpy.__version__)

1.2.0

[-]
'''import zipfile as zf

files = zf.ZipFile("../data/images/castings.zip", 'r')
files.extractall('../data/images/')
files.close()'''

[7]
from dlpy.images import ImageTable

[8]
img_path = '/data/casting_data/train/'
my_images = ImageTable.load_files(conn, path=img_path)
```

SAS Viya Deep Learning

[9]

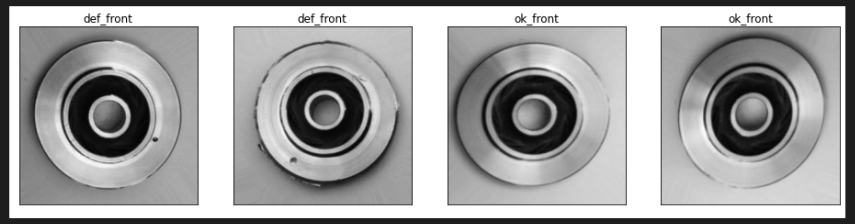
```
my_images.head()
```

Selected Rows from Table IMAGEDATA_SXJRKB

	image	_label_	_filename_0	_id_
0	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_8801.jpeg	5722
1	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	ok_front	cast_ok_0_1690.jpeg	2747
2	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_8172.jpeg	3712
3	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_5004.jpeg	6354
4	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_4932.jpeg	5724

[10]

```
my_images.show(nimages=4, ncol=4, randomize=True)
```



[11]

```
my_images.label_freq
```

Frequency for IMAGEDATA_SXJRKB

	Level	Frequency
def_front	1	3758
ok_front	2	2875

[12]

```
my_images.image_summary
```

```
jpg                 6633
minWidth             300
maxWidth             300
minHeight            300
maxHeight            300
meanWidth            300
meanHeight           300
mean1stChannel   143.878
min1stChannel        0
max1stChannel        255
mean2ndChannel   143.878
min2ndChannel        0
max2ndChannel        255
mean3rdChannel   143.878
min3rdChannel        0
max3rdChannel        255
dtype: object
```

SAS Viya Deep Learning

[13]

```
from dlpy.splitting import two_way_split

tr_img, te_img = two_way_split(my_images, test_rate=20, seed=123)
```

[14]

```
tr_img.head()
```

Selected Rows from Table TRAIN_3YA5PO

	image	_label_	_filename_0	_id_
0	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	ok_front	cast_ok_0_9438.jpeg	1232
1	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_6700.jpeg	3058
2	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_4744.jpeg	3430
3	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_6174.jpeg	4181
4	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_9175.jpeg	3168

[15]

```
te_img.head()
```

Selected Rows from Table TEST_WEZ0C7

	image	_label_	_filename_0	_id_
0	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_7693.jpeg	3695
1	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_9670.jpeg	6453
2	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_6914.jpeg	3966
3	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	def_front	cast_def_0_2648.jpeg	6492
4	b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00...	ok_front	cast_ok_0_3168.jpeg	1534

[16]

```
train_path = '/data/casting_data/train/'
test_path = '/data/casting_data/test/'
tr_img = ImageTable.load_files(conn, path=train_path)
te_img = ImageTable.load_files(conn, path=test_path)
```

[17]

```
tr_img.label_freq
```

Frequency for IMAGEDATA_JYDCTW

	Level	Frequency
def_front	1	3758
ok_front	2	2875

[18]

```
te_img.label_freq
```

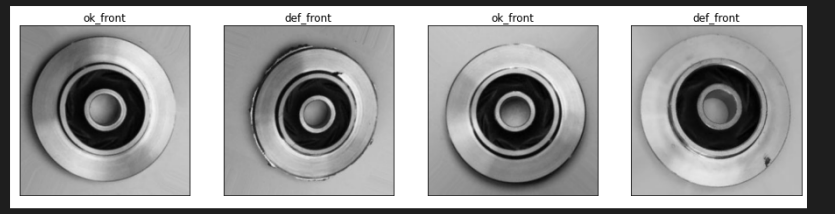
Frequency for IMAGEDATA_X7POH3

	Level	Frequency
def_front	1	453
ok_front	2	262

SAS Viya Deep Learning

[19]
```
tr_img.as_patches(width=300, height=300, step_size=24, output_width=300, output_height=300)
```

[20]
```
tr_img.show(4,4)
```



[21]
```
from dlpy import Sequential
from dlpy.model import *
from dlpy.layers import *
from dlpy.applications import *
```

[22]
```
# Create a sequential model named 'Simple CNN'
model1 = Sequential(conn, model_table = 'Simple_CNN')
```

[23]
```
# Specify input layer parameters for model1 'Simple CNN'
# Input layer has 3 channels (RGB), 300 px height and width
# Use offsets as defined by channel mean values for images
model1.add(InputLayer(3, 300, 300, offsets=tr_img.channel_means))
```

```
NOTE: Input layer added.
```

[24]
```
# Add 2D conv layer with 8 filters and kernel size 7
model1.add(Conv2d(8, 7))

# Add pooling layer of size 2
model1.add(Pooling(2))
```

```
NOTE: Convolution layer added.
NOTE: Pooling layer added.
```

[25]
```
# Add 2D conv layer with 8 filters and kernel size 7
model1.add(Conv2d(8,7))

# Add pooling layer of size 2
model1.add(Pooling(2))
```

```
NOTE: Convolution layer added.
NOTE: Pooling layer added.
```

[26]
```
# Add fully connected layer with 16 neurons
model1.add(Dense(16))
```

```
NOTE: Fully-connected layer added.
```

SAS Viya Deep Learning

[28]

```
model1.print_summary()
```

	Layer Id	Layer	Type	Kernel Size	Stride	Activation	Output Size	Number of Parameters	FLOPS(forward pass)
0	0	Input1	input			None	(300, 300, 3)	(0, 0)	0
1	1	Convo.1	convo	(7, 7)	(1, 1)	Relu	(300, 300, 8)	(1176, 8)	105840000
2	2	Pool1	pool	(2, 2)	(2, 2)	Max	(150, 150, 8)	(0, 0)	0
3	3	Convo.2	convo	(7, 7)	(1, 1)	Relu	(150, 150, 8)	(3136, 8)	70560000
4	4	Pool2	pool	(2, 2)	(2, 2)	Max	(75, 75, 8)	(0, 0)	0
5	5	F.C.1	fc	(45000, 16)		Relu	16	(720000, 0)	720000
6	6	Output1	output			Softmax	2	(32, 2)	0
7								Total number of parameters	Total FLOPS
8	Summary							724,362	177,120,000

[29]

```
model1.fit(data=tr_img,
           mini_batch_size=2,
           max_epochs=30,
           lr=1E-4,
           gpu=Gpu(devices=[1]),
           log_level=2)
```

```
NOTE: Inputs=_image_ is used
NOTE: Training from scratch.
WARNING: Worker with rank 0 has no GPU.
WARNING: Worker with rank 1 has no GPU.
NOTE:  Synchronous mode is enabled.
NOTE:  The total number of parameters is 724378.
NOTE:  The approximate memory cost is 476.00 MB.
NOTE:  Loading weights cost       0.00 (s).
NOTE:  Initializing each layer cost       0.23 (s).
NOTE:  The total number of workers is 2.
NOTE:  The total number of threads on each worker is 4.
NOTE:  The total mini-batch size per thread on each worker is 2.
NOTE:  The maximum mini-batch size across all workers for the synchronous mode is 16.
NOTE:  Target variable: _label_
NOTE:  Number of levels for the target variable:      2
NOTE:  Levels for the target variable:
NOTE:  Level      0: def_front
NOTE:  Level      1: ok_front
NOTE:  Number of input variables:     1
NOTE:  Number of numeric input variables:      1
NOTE:  Epoch Learning Rate        Loss  Fit Error   Time(s)
NOTE:  0        0.0001           0.477      0.198    94.93
NOTE:  1        0.0001          0.2418     0.1023    94.29
NOTE:  2        0.0001          0.1845    0.07007    93.81
```

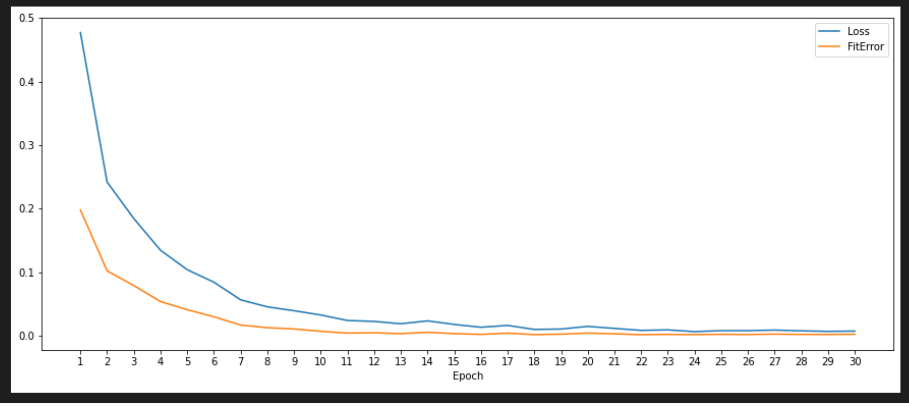
SAS Viya Deep Learning

- Table of the training history of the model.
- After 30 epochs, the loss is reduced to approximately 0.007441.

```
[30]
# Generate model training history table
model1.training_history
```

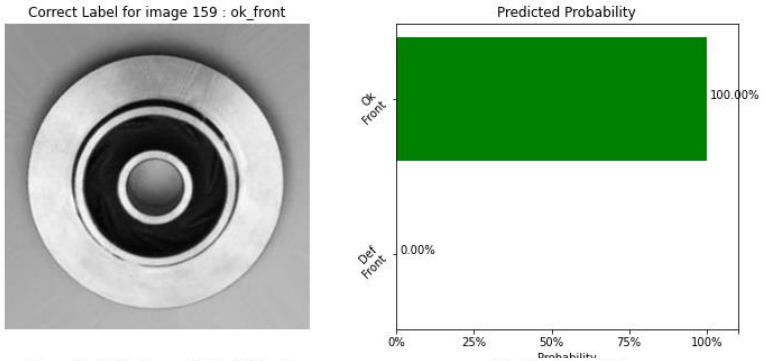
	Epoch	LearningRate	Loss	FitError
0	1	0.0001	0.477026	0.198042
1	2	0.0001	0.241765	0.102259
2	3	0.0001	0.184523	0.079066
3	4	0.0001	0.134593	0.053916
4	5	0.0001	0.104062	0.041114
5	6	0.0001	0.084345	0.029970
6	7	0.0001	0.056675	0.017018
7	8	0.0001	0.045594	0.012801
8	9	0.0001	0.039505	0.010843
9	10	0.0001	0.032812	0.007078
10	11	0.0001	0.024144	0.004367
11	12	0.0001	0.022686	0.004819
12	13	0.0001	0.019003	0.003464
13	14	0.0001	0.023528	0.005572
14	15	0.0001	0.017939	0.003464
15	16	0.0001	0.013471	0.002259
16	17	0.0001	0.016377	0.004217
17	18	0.0001	0.010086	0.001958
18	19	0.0001	0.010665	0.002560
19	20	0.0001	0.014836	0.004367
20	21	0.0001	0.011714	0.003163
21	22	0.0001	0.008587	0.001807
22	23	0.0001	0.009359	0.002259
23	24	0.0001	0.006621	0.001958
24	25	0.0001	0.008144	0.002410
25	26	0.0001	0.008193	0.002108
26	27	0.0001	0.009051	0.002711
27	28	0.0001	0.007832	0.002259
28	29	0.0001	0.007025	0.002108
29	30	0.0001	0.007441	0.002560

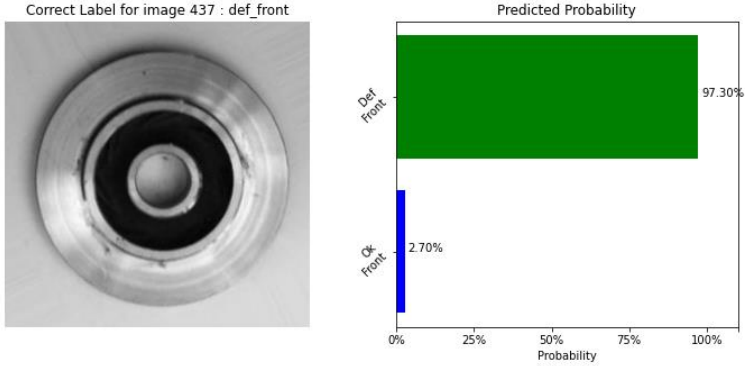
```
[31]
# Plot data in training history table
model1.plot_training_history(fig_size=(15,6))

<matplotlib.axes._subplots.AxesSubplot at 0x7f4a8a044390>
```



SAS Viya Deep Learning

- Model is saved and used to score the test data.
- 1.26% Misclassification Error






```
[32]
# Score model predictions for test data
model1.evaluate(te_img)
```

§ ScoreInfo

	Descr	Value
0	Number of Observations Read	715
1	Number of Observations Used	715
2	Misclassification Error (%)	1.258741
3	Loss Error	0.036705

§ OutputCasTables

	casLib	Name	Rows	Columns	casTable
0	CASUSERHDFS(scott.koval)	Valid_Res_StjV1Q	715	9	CASTable('Valid_Res_StjV1Q', caslib='CASUSERHD...

elapsed 4.54s · user 31.7s · sys 0.437s · mem 1.38e+03MB

```
[33]
model1.valid_conf_mat
```

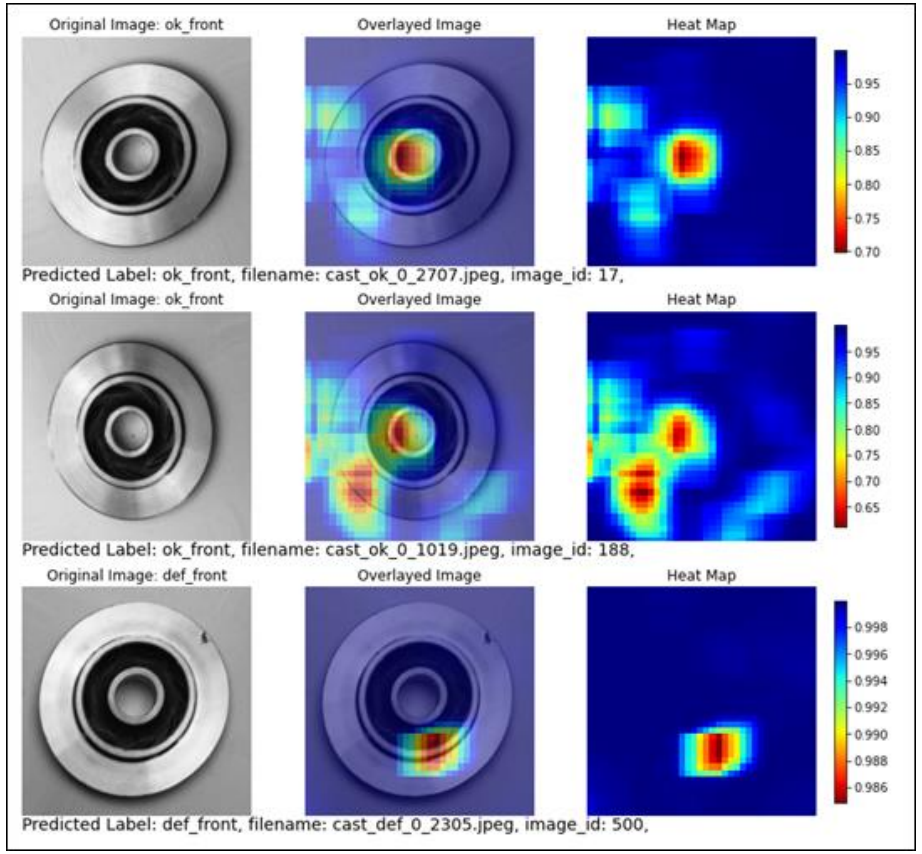
§ Crosstab

	label	Col1	Col2
0	def_front	445.0	8.0
1	ok_front	1.0	261.0

elapsed 0.0158s · user 0.00776s · sys 0.0136s · mem 2.98MB

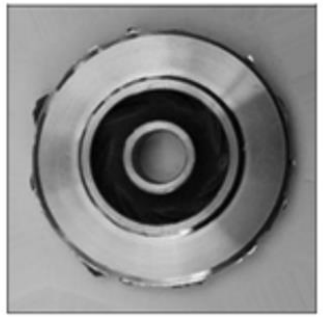
SAS Viya Deep Learning

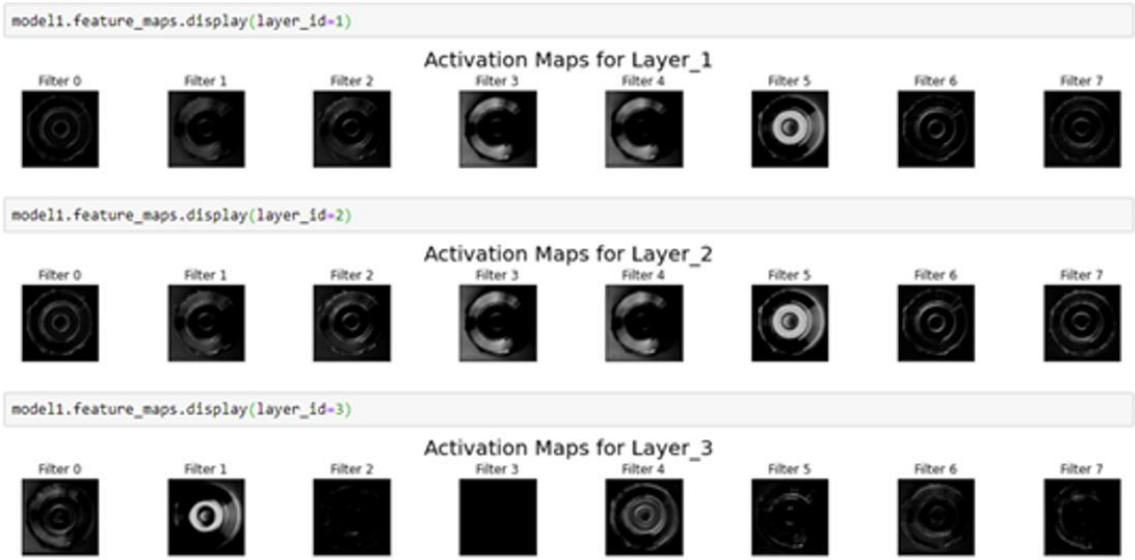
- We can try to interpret the final model using heatmaps.



SAS Viya Deep Learning

- Can also investigate what is being activated for individual layers of the CNN.
- The final model can be saved as an astore file and used to score new images.





Overview

- The SAS Viya platform allows for powerful analytics.
  - Access to many methods to develop ML and AI models.
  - SAS maintains a useful GitHub repository with example code.
- Can be scalable and deployed at the edge.

Thank You!

- Scott Koval
- Pinnacle Solutions, Inc
- scott.koval@thepinnaclesolutions.com