기계학습과 응용

Group 10

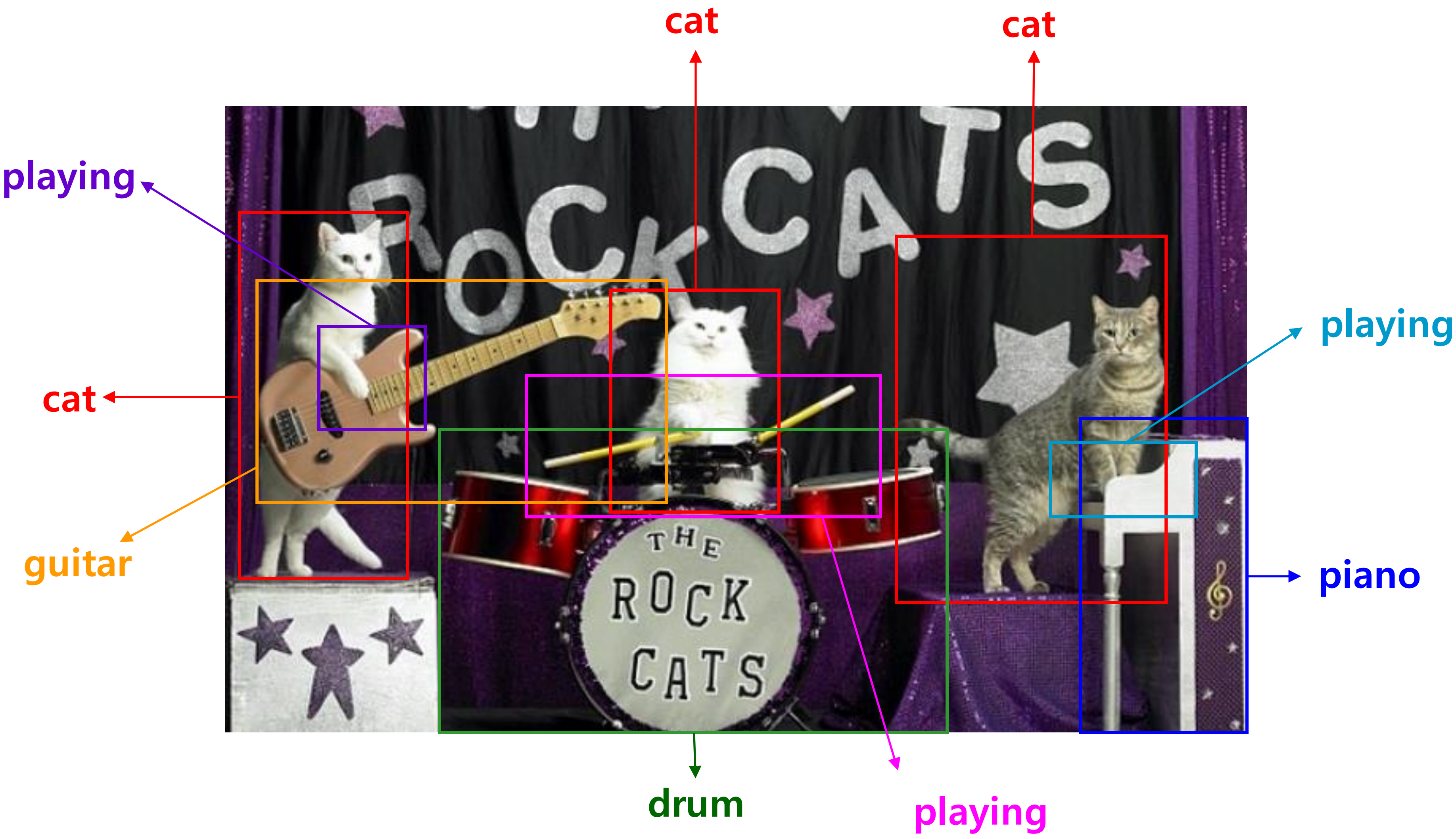
김상훈 변상준 심재헌 정희명

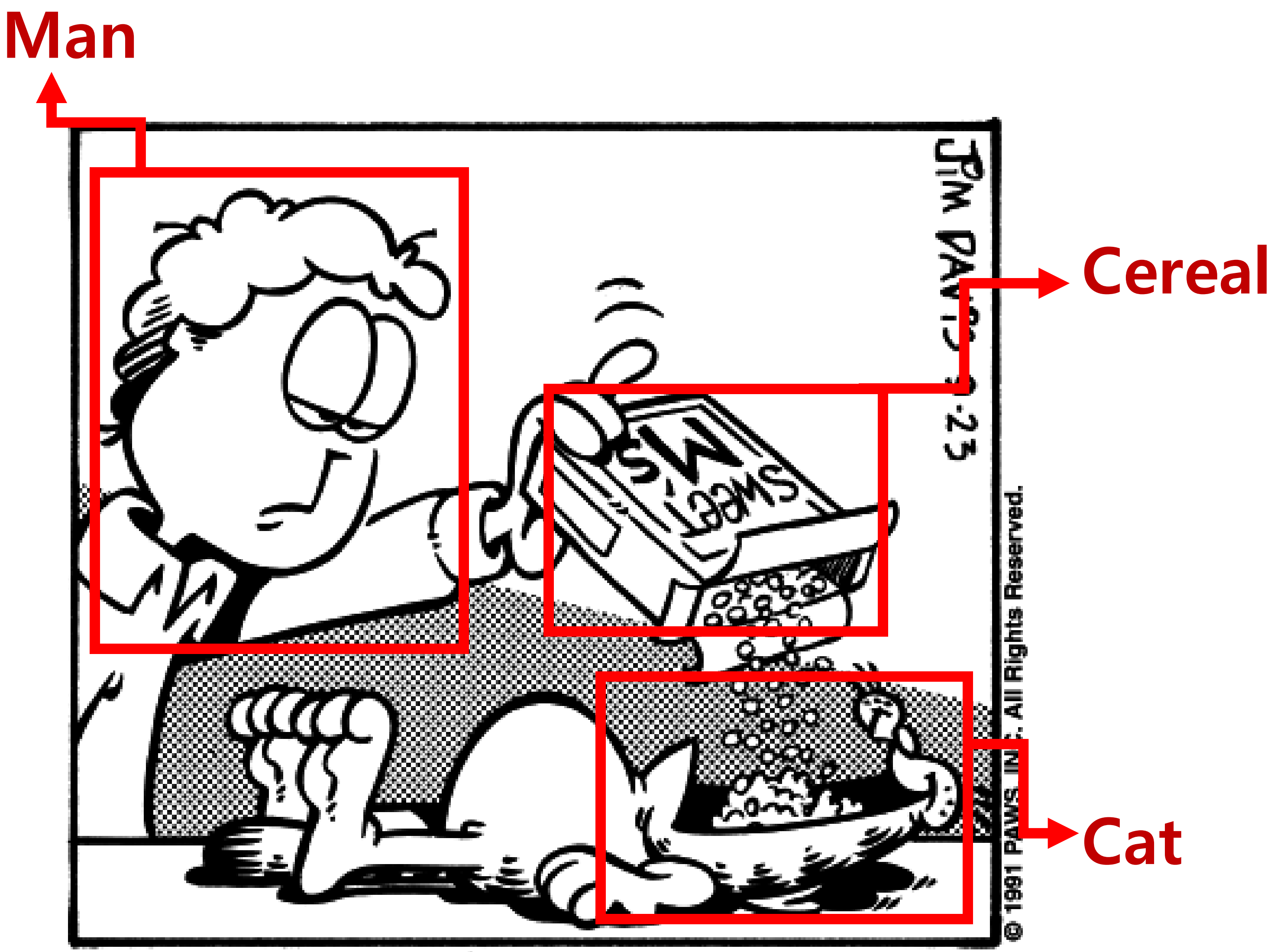
Index

Idea

Idea

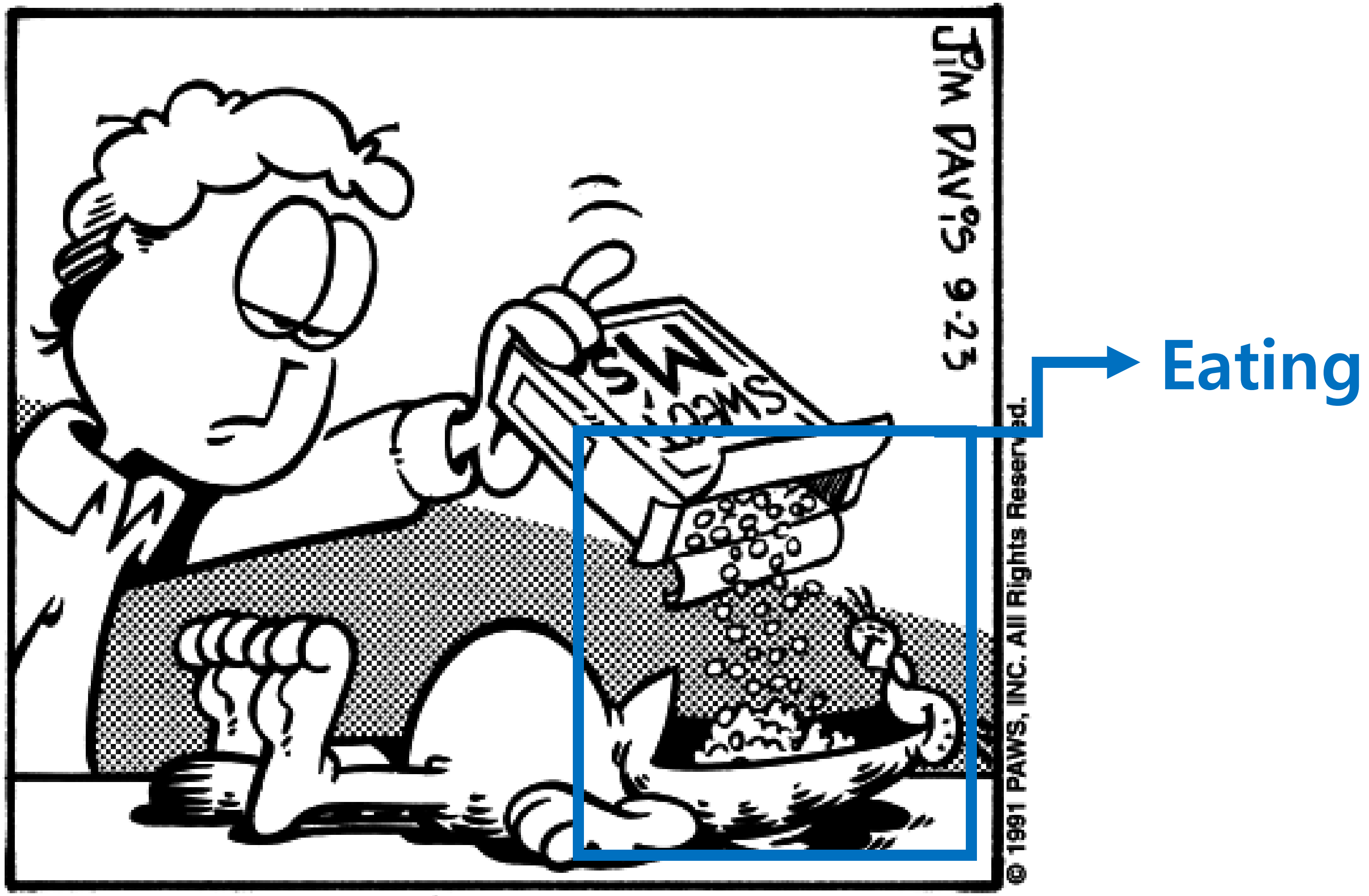
What's our goal?





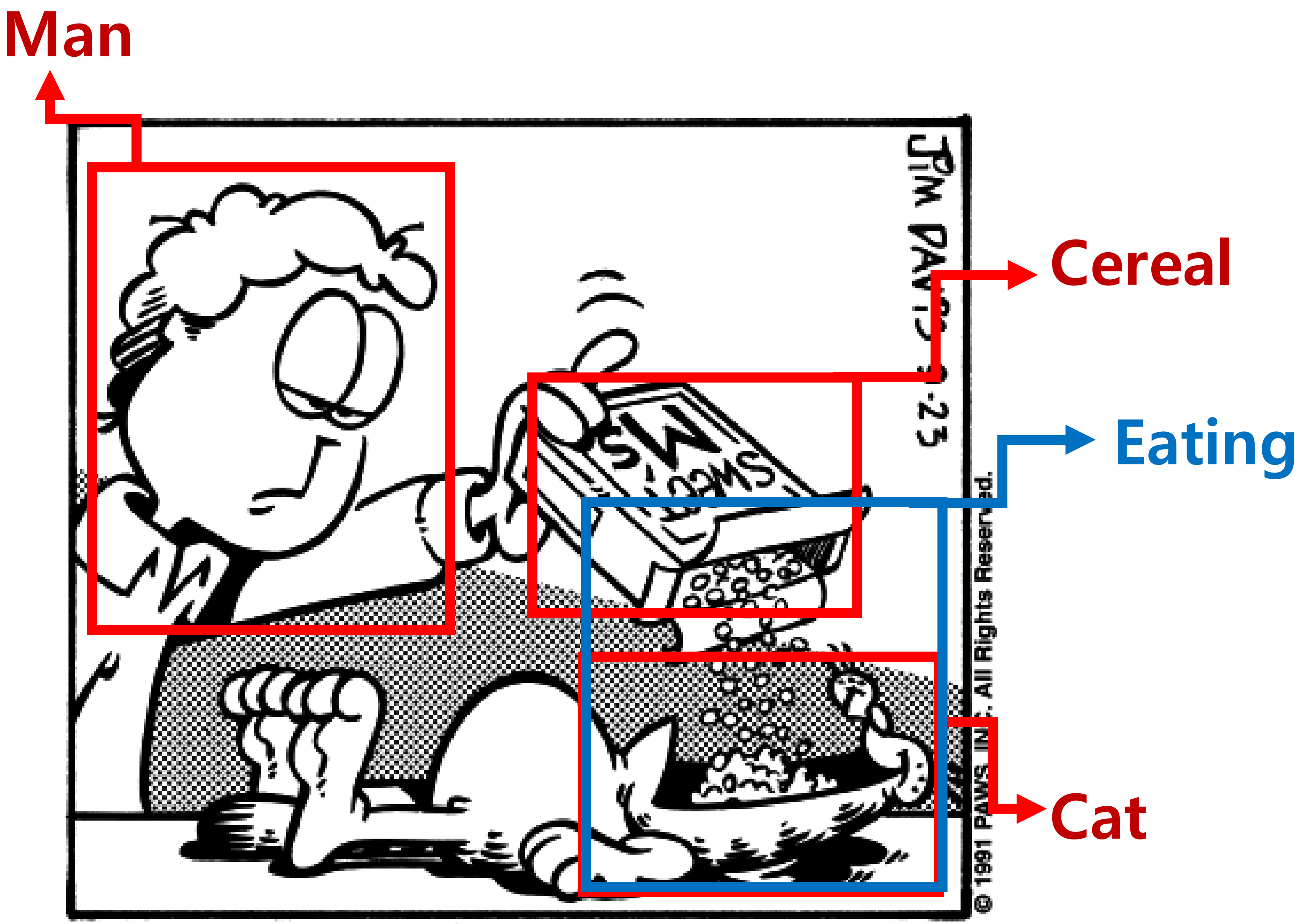
Object Detection

1. Man
2. Cereal
3. Cat



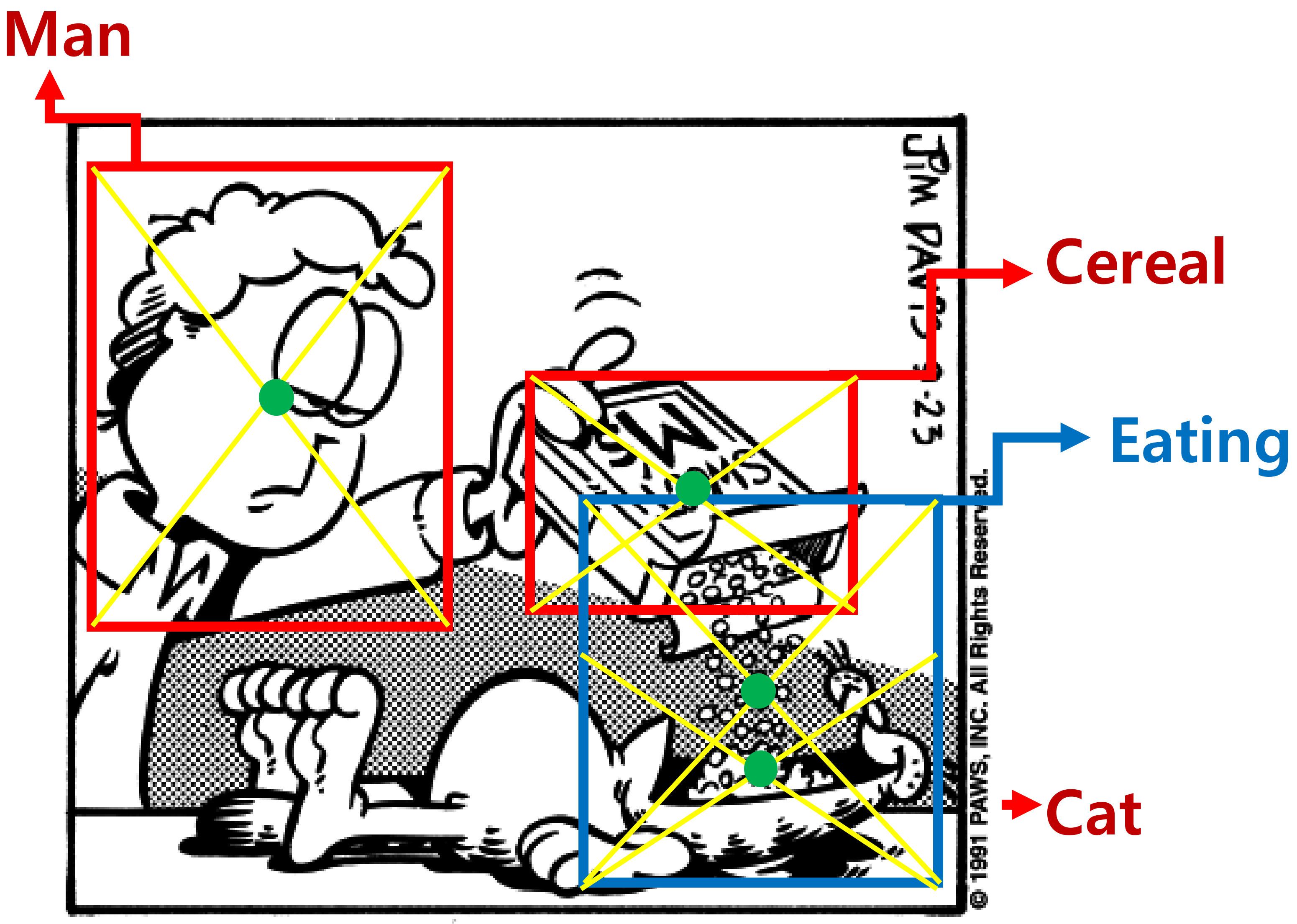
Action Detection

1. Eating



Result

1. Man
2. Cereal
3. Cat
4. Eating

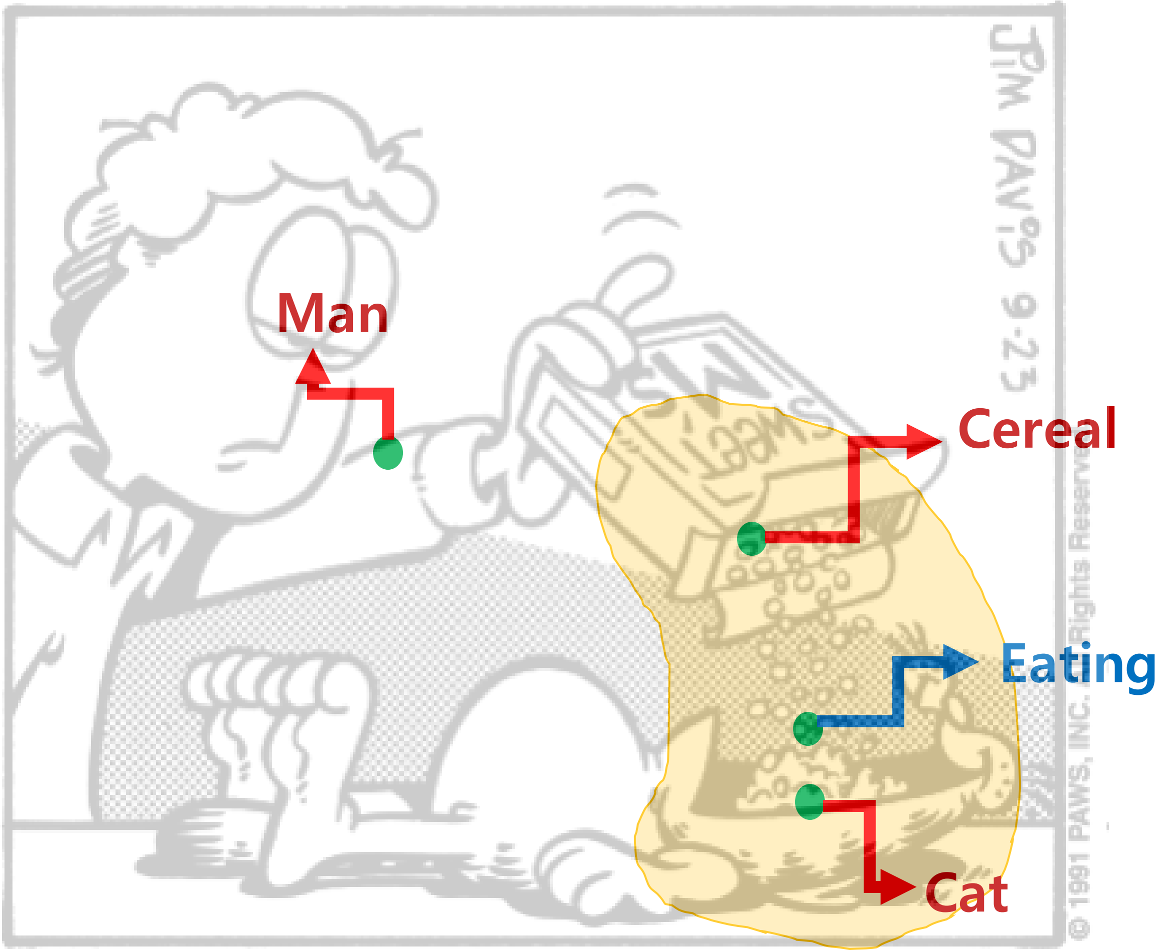


Grouping

1. Man
2. Cereal
3. Cat
4. Eating

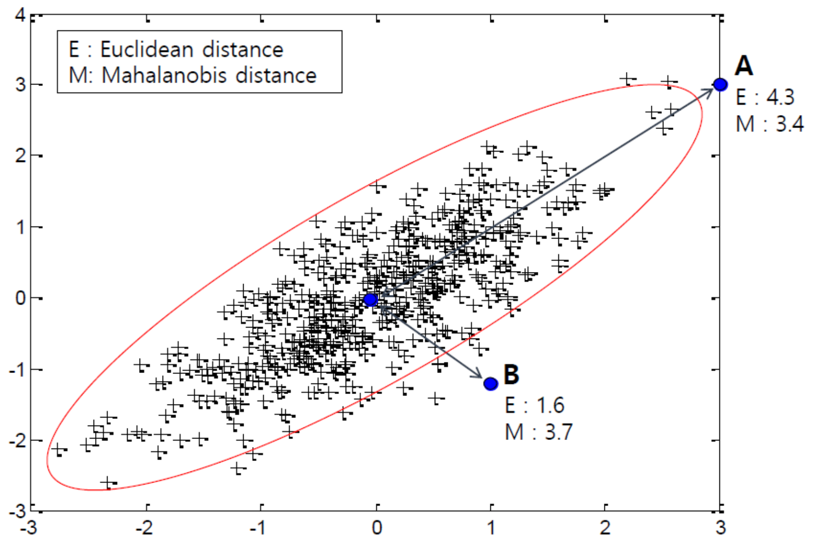
Method

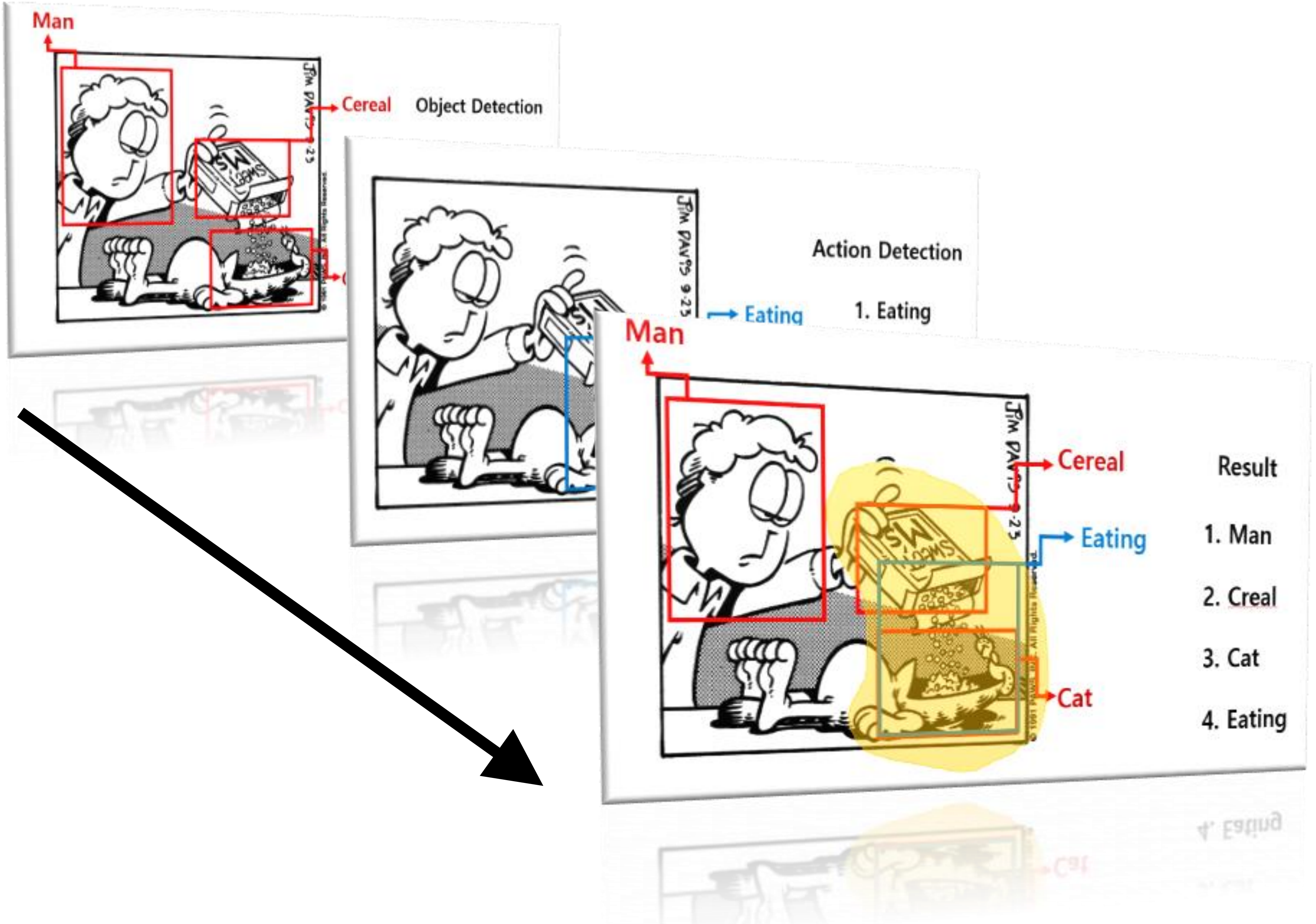
K-Nearest Neighbor(KNN)



$$d_{Mahalanobis(X,Y)} = \sqrt{(\vec{X} - \vec{Y})^T \Sigma^{-1} (\vec{X} - \vec{Y})}$$

$$\Sigma^{-1} = inverse \quad of \quad covariance \quad matrix$$



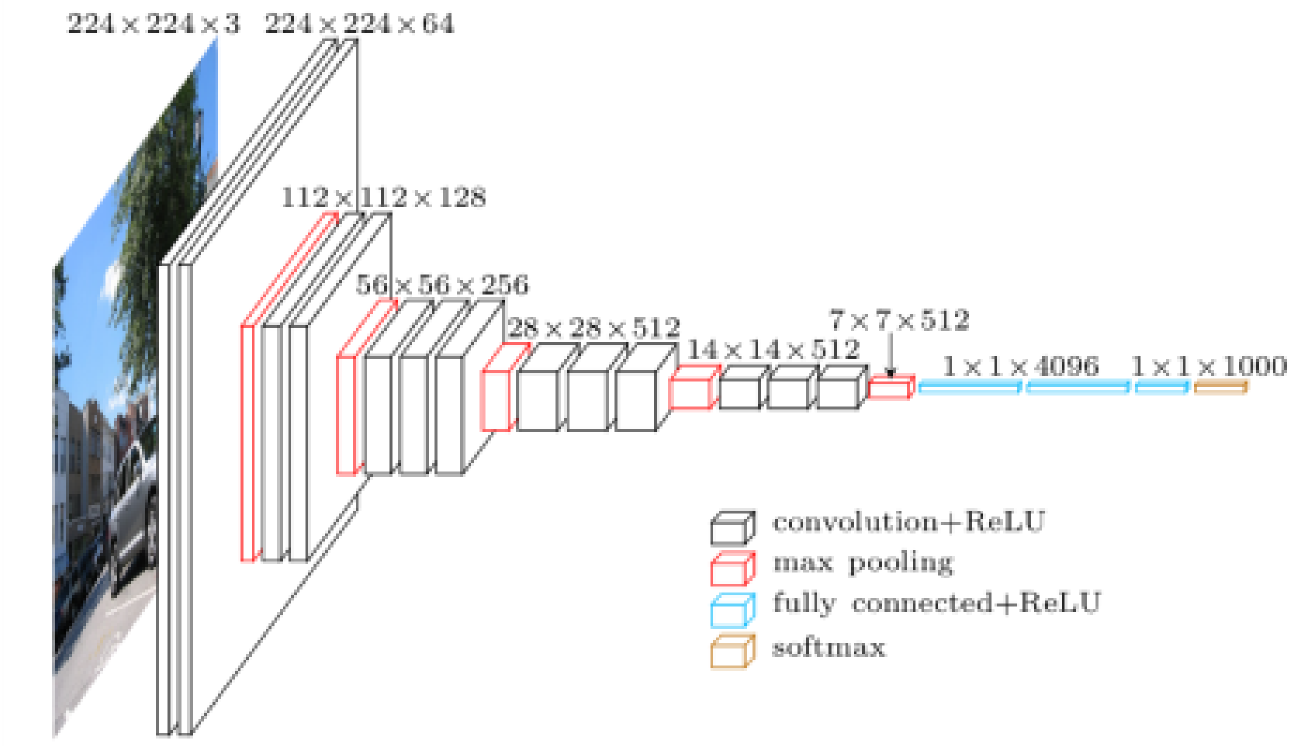


Man
Cereal
Object Detection
Action Detection
Eating
1. Eating
Man
Cereal
Eating
Cat
Result
1. Man
2. Creal
3. Cat
4. Eating

Method

Method

Convolutional Neural Network (CNN)



Convolution

↓

Max pooling

↓

Convolution

↓

Max pooling

↓

Convolution

↓

Max pooling

↓

Fully connected

↓

Softmax regression

CNN Algorithm Comparison

- There are many algorithm models: **ResNet**, **InceptionV3**, DenseNet, SqueezeNet etc.

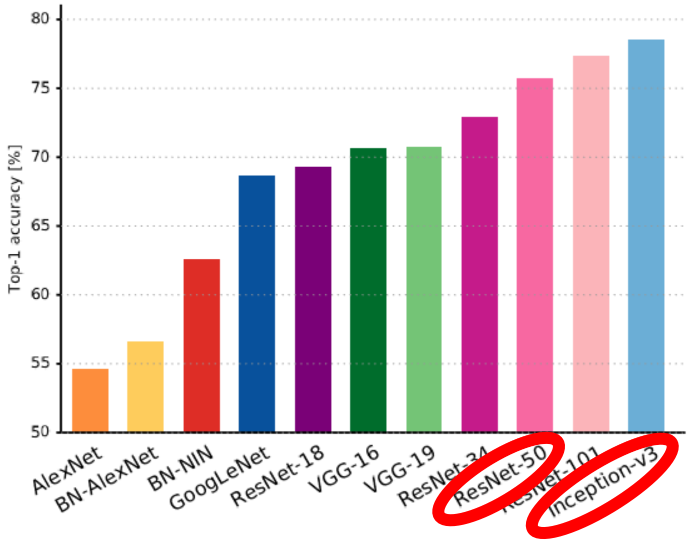


Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that network of the same group share colour, for example ResNet are all variations of pink.

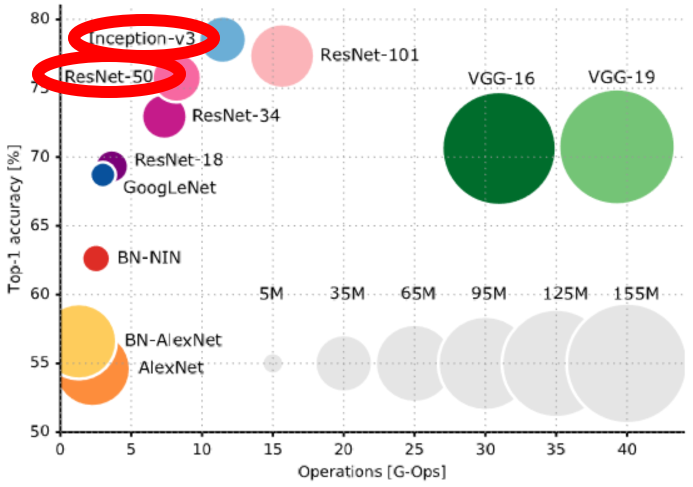


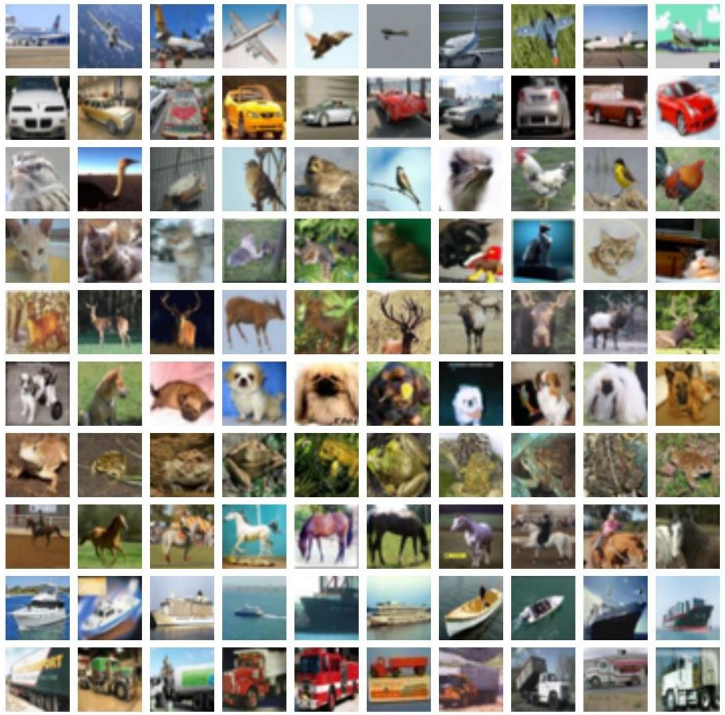
Figure 2: **Top1 *vs.* operations, size and parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params.

Modeling

Modeling

Training Dataset

CIFAR-10 Datasets



Dog(200)

Cat(200)

Training

ResNet-50

InceptionV3

Modeling

Training Models(ResNet, Inception-V3)

```
# Training ResNet Model

from imageai.Prediction.Custom import ModelTraining

model_trainer = ModelTraining()
model_trainer.setModelTypeAsResNet()
model_trainer.setDataDirectory("jaeheon/colab/pets")
model_trainer.trainModel(num_objects=2, num_experiments=100, enhance_data=True, batch_size=25, show_network_summary=True)
```

ResNet

```
# Training Inception-V3 Model

from imageai.Prediction.Custom import ModelTraining


model_trainer = ModelTraining()
model_trainer.setModelTypeAsInceptionV3()
model_trainer.setDataDirectory("pets")
model_trainer.trainModel(num_objects=2, num_experiments=100, enhance_data=True, batch_size=25, show_network_summary=True)
```

Inception-V3

Modeling

Training

In [*]:
```python
from imageai.Prediction.Custom import ModelTraining



model_trainer = ModelTraining()
model_trainer.setModelTypeAsInceptionV3()
model_trainer.setDataDirectory("pets")
model_trainer.trainModel(num_objects=2, num_experiments=100, enhance_data=True, batch_size=25, show_network_summary=True)
```

```
16/16 [==========================] - 544s 34s/step - loss: 0.3407 - acc: 0.8525 - val_loss: 0.9190 - val_acc: 0.6400
Epoch 26/100
15/16 [========================>..] - ETA: 32s - loss: 0.3112 - acc: 0.8747
Epoch 00026: saving model to pets₩models₩model_ex-026_acc-0.620000.h5
16/16 [==========================] - 546s 34s/step - loss: 0.3110 - acc: 0.8725 - val_loss: 1.2728 - val_acc: 0.6200
Epoch 27/100
15/16 [========================>..] - ETA: 53s - loss: 0.3760 - acc: 0.8480
Epoch 00027: saving model to pets₩models₩model_ex-027_acc-0.660000.h5
16/16 [==========================] - 898s 56s/step - loss: 0.3756 - acc: 0.8525 - val_loss: 0.9130 - val_acc: 0.6600
Epoch 28/100
15/16 [========================>..] - ETA: 38s - loss: 0.3092 - acc: 0.8693
Epoch 00028: saving model to pets₩models₩model_ex-028_acc-0.570000.h5
16/16 [==========================] - 629s 39s/step - loss: 0.3142 - acc: 0.8675 - val_loss: 2.1252 - val_acc: 0.5700
Epoch 29/100
15/16 [========================>..] - ETA: 39s - loss: 0.2817 - acc: 0.8827
Epoch 00029: saving model to pets₩models₩model_ex-029_acc-0.580000.h5

16/16 [==========================] - 662s 41s/step - loss: 0.2949 - acc: 0.8725 - val_loss: 1.8313 - val_acc: 0.5800
Epoch 30/100
 1/16 [>.............................] - ETA: 11:14 - loss: 0.1012 - acc: 0.9600
```

Conclusion

Conclusion

ResNet-50

```
15/16 [===========================>..] - ETA: 3s - loss: 0.1892 - acc: 0.9173
Epoch 00096: saving model to jaeheon/colab/pets/models/model_ex-096_acc-0.690000.h5
16/16 [==============================] - 88s 6s/step - loss: 0.1811 - acc: 0.9225 - val_loss: 0.9801 - val_acc: 0.6900
Epoch 97/100
15/16 [===========================>..] - ETA: 2s - loss: 0.2092 - acc: 0.9173
Epoch 00097: saving model to jaeheon/colab/pets/models/model_ex-097_acc-0.690000.h5
16/16 [==============================] - 76s 5s/step - loss: 0.2163 - acc: 0.9150 - val_loss: 0.9766 - val_acc: 0.6900
Epoch 98/100
15/16 [===========================>..] - ETA: 4s - loss: 0.2069 - acc: 0.9173
Epoch 00098: saving model to jaeheon/colab/pets/models/model_ex-098_acc-0.690000.h5
16/16 [==============================] - 102s 6s/step - loss: 0.2026 - acc: 0.9175 - val_loss: 0.9805 - val_acc: 0.6900
Epoch 99/100
15/16 [===========================>..] - ETA: 2s - loss: 0.1624 - acc: 0.9333
Epoch 00099: saving model to jaeheon/colab/pets/models/model_ex-099_acc-0.690000.h5
16/16 [==============================] - 69s 4s/step - loss: 0.1596 - acc: 0.9350 - val_loss: 0.9809 - val_acc: 0.6900
Epoch 100/100
15/16 [===========================>..] - ETA: 2s - loss: 0.2092 - acc: 0.9253
Epoch 00100: saving model to jaeheon/colab/pets/models/model_ex-100_acc-0.690000.h5
16/16 [==============================] - 52s 3s/step - loss: 0.2029 - acc: 0.9250 - val_loss: 0.9815 - val_acc: 0.6900
```

Conclusion

InceptionV3

```
Epoch 75/100
15/16 [========================>..] - ETA: 34s - loss: 0.0174 - acc: 0.9947
Epoch 00075: saving model to pets₩models₩model_ex-075_acc-0.790000.h5
16/16 [==========================] - 581s 36s/step - loss: 0.0171 - acc: 0.9950 - val_loss: 0.8029 - val_acc: 0.7900
Epoch 76/100
15/16 [========================>..] - ETA: 36s - loss: 0.0260 - acc: 0.9920
Epoch 00076: saving model to pets₩models₩model_ex-076_acc-0.790000.h5
16/16 [==========================] - 617s 39s/step - loss: 0.0248 - acc: 0.9925 - val_loss: 0.8083 - val_acc: 0.7900
Epoch 77/100
15/16 [========================>..] - ETA: 41s - loss: 0.0164 - acc: 0.9947
Epoch 00077: saving model to pets₩models₩model_ex-077_acc-0.800000.h5
16/16 [==========================] - 705s 44s/step - loss: 0.0195 - acc: 0.9950 - val_loss: 0.8113 - val_acc: 0.8000
Epoch 78/100
15/16 [========================>..] - ETA: 34s - loss: 0.0116 - acc: 1.0000
Epoch 00078: saving model to pets₩models₩model_ex-078_acc-0.800000.h5
16/16 [==========================] - 565s 35s/step - loss: 0.0118 - acc: 1.0000 - val_loss: 0.8108 - val_acc: 0.8000
Epoch 79/100
15/16 [========================>..] - ETA: 34s - loss: 0.0292 - acc: 0.9893
Epoch 00079: saving model to pets₩models₩model_ex-079_acc-0.780000.h5
16/16 [==========================] - 581s 36s/step - loss: 0.0287 - acc: 0.9900 - val_loss: 0.8115 - val_acc: 0.7800
```

Conclusion

InceptionV3

```
from imageai.Prediction.Custom import CustomImagePrediction
import os
execution_path = os.getcwd()


prediction = CustomImagePrediction()
prediction.setModelTypeAsInceptionV3()
prediction.setModelPath(os.path.join(execution_path, "model_ex-078_acc-0.800000.h5"))
prediction.setJsonPath(os.path.join(execution_path, "model_class.json"))
prediction.loadModel(num_objects=2)


predictions, probabilities = prediction.predictImage(os.path.join(execution_path, "5.jpg"), result_count=2)


for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction + " : " + eachProbability)
```

```
cat : 99.99308586120605
dog : 0.006908018985996023
```

Conclusion

InceptionV3

```
from imageai.Prediction.Custom import CustomImagePrediction
import os
execution_path = os.getcwd()


prediction = CustomImagePrediction()
prediction.setModelTypeAsInceptionV3()
prediction.setModelPath(os.path.join(execution_path, "model_ex-078_acc-0.800000.h5"))
prediction.setJsonPath(os.path.join(execution_path, "model_class.json"))
prediction.loadModel(num_objects=2)


predictions, probabilities = prediction.predictImage(os.path.join(execution_path, "1.jpg"), result_count=2)


for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction + " : " + eachProbability)
```

```
dog : 99.02809858322144
cat : 0.9719012305140495
```

THANK YOU