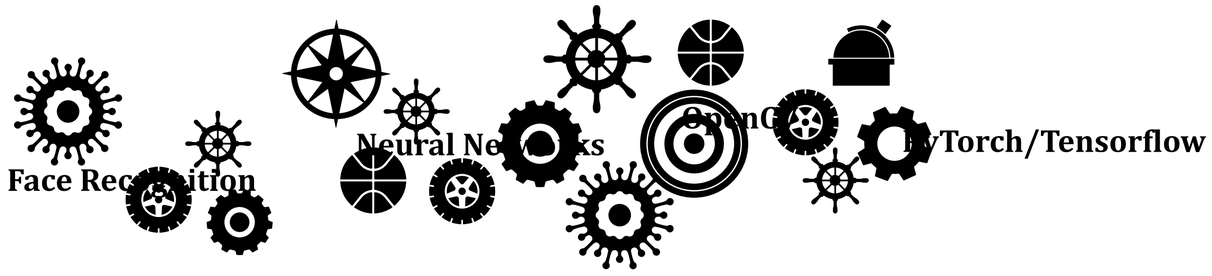
MPSSDEGB
कौशल से रोजगार, समृद्धि अपार

FOUNDATION FOR INNOVATION AND TECHNOLOGY TRANSFER
भारतीय प्रौद्योगिकी संस्थान दिल्ली
Indian Institute of Technology Delhi

Computer Vision



Deep Learning
Segmentation
Object Detection
Image Classification

Recap

- We defined a *Bayes classifier* but saw that it's intractable to compute $P(X_1,\ldots,X_n|Y)$
- We then used the *Naïve Bayes assumption* – that everything is independent given the class label Y

- Naïve Bayes is:
  - Really easy to implement and often works well
  - Often a good first thing to try
  - Commonly used as a "punching bag" for smarter algorithms

Decision Trees

Introduction

- A classification scheme which generates a tree and a set of rules from given data set.
- The set of records available for developing classification methods is divided into two disjoint subsets – a *training set* and a *test set*.
- The attributes of the records are categorise into two types:
  - Attributes whose domain is numerical are called numerical attributes.
  - Attributes whose domain is not numerical are called the categorical attributes.

https://www.slideshare.net/RajendraAkerkar/decision-tree-9146388

Decision Trees

- *A decision tree* is a tree with the following properties:
  - An inner node represents an attribute.
  - An edge represents a test on the attribute of the father node.
  - A leaf represents one of the classes.
- Construction of a decision tree
  - Based on the training data
  - Top-Down strategy

Decision Trees

Decision Tree

Example

Table 6.1 Training Data Set

OUTLOOK	TEMP(F)	HUMIDITY(%)	WINDY	CLASS
sunny	79	90	true	no play
sunny	56	70	false	play
sunny	79	75	true	play
sunny	60	90	true	no play
overcast	88	88	false	no play
overcast	63	75	true	play
overcast	88	95	false	play
rain	78	60	false	play
rain	66	70	false	no play
rain	68	60	true	no play

- The data set has five attributes.
- There is a special attribute: the attribute *class* is the class label.
- The attributes, *temp* (temperature) and *humidity* are numerical attributes
- Other attributes are categorical, that is, they cannot be ordered.

- Based on the training data set, we want to find a set of rules to know what values of *outlook, temperature, humidity* and *wind,* determine whether or not to play golf.

Decision Trees

Decision Tree

Example

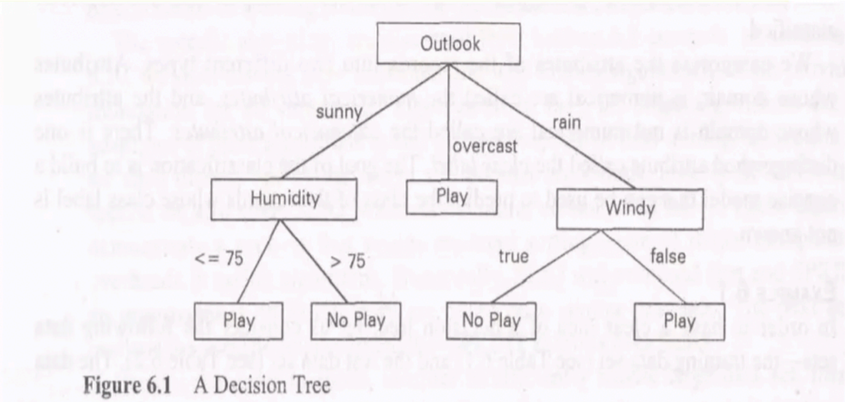


Figure 6.1 A Decision Tree

- We have five leaf nodes.
- In a decision tree, each leaf node represents a rule.

- We have the following rules corresponding to the tree given in Figure.

- *RULE 1* *If it is sunny and the humidity is not above 75%, then play.*
- *RULE 2* *If it is sunny and the humidity is above 75%, then do not play.*
- *RULE 3* *If it is overcast, then play.*
- *RULE 4* *If it is rainy and not windy, then play.*
- *RULE 5* *If it is rainy and windy, then don't play.*

Decision Trees

Classification

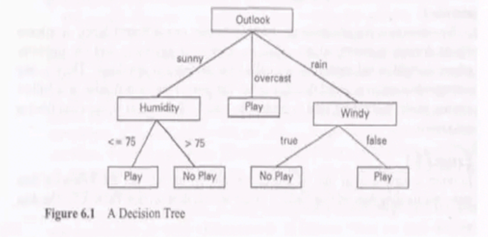


Figure 6.1 A Decision Tree

- The classification of an unknown input vector is done by traversing the tree from the root node to a leaf node.
- A record enters the tree at the root node.
- At the root, a test is applied to determine which child node the record will encounter next.
- This process is repeated until the record arrives at a leaf node.
- All the records that end up at a given leaf of the tree are classified in the same way.
- There is a unique path from the root to each leaf.
- The path is a rule which is used to classify the records.

Decision Trees

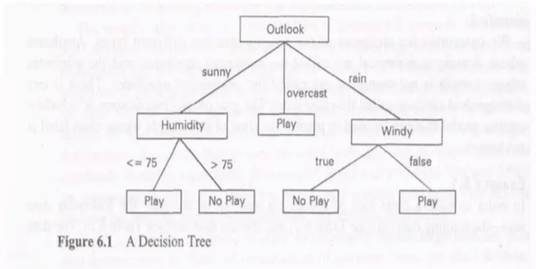


Figure 6.1 A Decision Tree

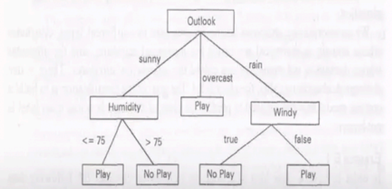
- In our tree, we can carry out the classification for an unknown record as follows.
- Let us assume, for the record, that we know the values of the first four attributes (but we do not know the value of *class* attribute) as

- *outlook= rain; temp = 70; humidity = 65; and windy= true.*

Decision Trees

- We start from the root node to check the value of the attribute associated at the root node.
- This attribute is the *splitting attribute* at this node.
- For a decision tree, at every node there is an attribute associated with the node called the *splitting attribute.*

- In our example, *outlook* is the splitting attribute at root.
- Since for the given record, *outlook* = *rain,* we move to the right-most child node of the root.
- At this node, the splitting attribute is *windy* and we find that for the record we want classify, *windy* = *true.*
- Hence, we move to the left child node to conclude that the class label Is "*no play*".



https://www.slideshare.net/RajendraAkerkar/decision-tree-9146388

Decision Trees

- The *accuracy of the classifier* is determined by the percentage of the test data set that is correctly classified.

- We can see that for *Rule 1* there are two records of the test data set satisfying *outlook= sunny* and *humidity < 75,* and only one of these is correctly classified *as play.*
- Thus, the accuracy of this rule is 0.5 (or 50%). Similarly, the accuracy of Rule 2 is also 0.5 (or 50%). The accuracy of Rule 3 is 0.66.

RULE 1
If it is sunny and the humidity is not above 75%, then play.

Table 6.2 Test Data Set

OUTLOOK	TEMP(F)	HUMIDITY(%)	WINDY	CLASS
sunny	79	90	true	play
sunny	56	70	false	play
sunny	79	75	true	no play
sunny	60	90	true	no play
overcast	88	88	false	no play
overcast	63	75	true	play
overcast	88	95	false	play
rain	78	60	false	play
rain	66	70	false	no play
rain	68	60	true	play

Decision Trees

Concept of Categorical Attributes

- Consider the following training data set.
- There are three attributes, namely, *age, pincode* and *class.*
- The attribute *class* is used for class label.

Table 6.3 Another Example

ID	AGE	PINCODE	CLASS
1	30	5600046	C1
2	25	5600046	C1
3	21	5600023	C2
4	43	5600046	C1
5	18	5600023	C2
6	33	5600023	C1
7	29	5600023	C1
8	55	5600046	C2
9	48	5600046	C1

The attribute *age* is a numeric attribute, whereas *pincode* is a categorical one.

Though the domain of *pincode* is numeric, no ordering can be defined among *pincode* values.

You cannot derive any useful information if one pin-code is greater than another *pincode*.

Decision Trees

- Figure gives a decision tree for the training data.
- The splitting attribute at the root is *pincode* and the splitting criterion here is *pincode = 500 046.*
- Similarly, for the left child node, the splitting criterion is *age < 48* (the splitting attribute is *age).*
- Although the right child node has the same attribute as the splitting attribute, the splitting criterion is different.

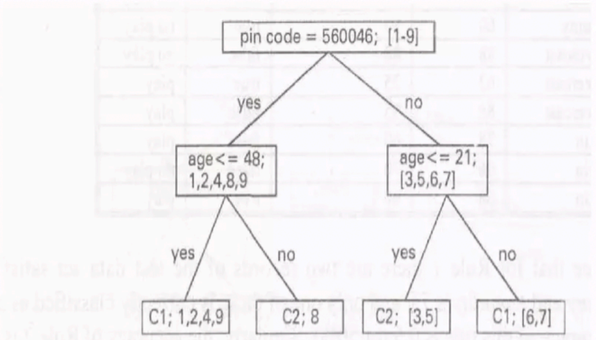


Figure 6.2 A Decision Tree

At root level, we have 9 records. The associated splitting criterion is *pincode* = 500 046.

As a result, we split the records into two subsets. Records 1, 2, 4, 8, and 9 are to the left child note and remaining to the right node.

The process is repeated at every node.

Decision Trees

Advantages and Shortcomings of Decision Tree Classifications

- A decision tree construction process is concerned with identifying the splitting attributes and splitting criterion at every level of the tree.
- Major strengths are:
  - Decision tree able to generate understandable rules.
  - They are able to handle both numerical and categorical attributes.
  - They provide clear indication of which fields are most important for prediction or classification.
- Weaknesses are:
  - The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field is examined before its best split can be found.
  - Some decision tree can only deal with binary-valued target classes.

https://www.slideshare.net/RajendraAkerkar/decision-tree-9146388

- At each node the splitting attribute is selected to be the most informative among the attributes not yet considered in the path from the root.

- *Entropy* is used to measure how informative is a node.
- The algorithm uses the criterion of *information gain* to determine the goodness of a split.
  - The attribute with the greatest information gain is taken as the splitting attribute, and the data set is split for all distinct values of the attribute.

Training Dataset

The class label attribute, *buys_computer*, has two distinct values.

Thus there are two distinct classes. (m =2)

Class C1 corresponds to *yes* and class C2 corresponds to *no*.

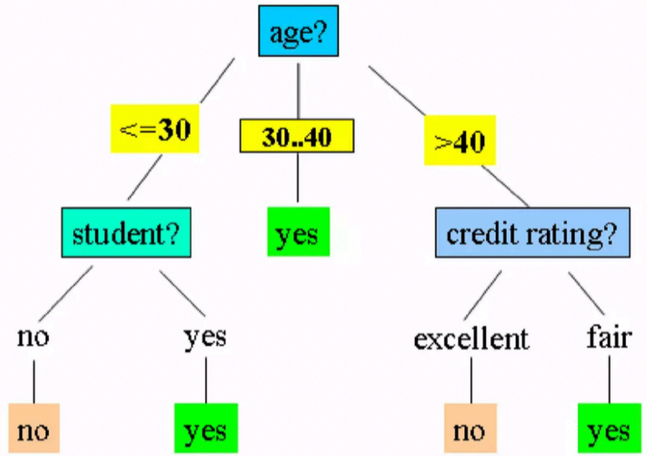
There are 9 samples of class *yes* and 5 samples of class *no*.

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Decision Trees

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand



What are the rules?

Decision Trees

Solution (Rules)

IF *age* = “<=30” AND *student* = “*no*” THEN *buys_computer* = “*no*”

IF *age* = “<=30” AND *student* = “*yes*” THEN *buys_computer* = “*yes*”

IF *age* = “31…40” THEN *buys_computer* = “*yes*”

IF *age* = “>40” AND *credit_rating* = “*excellent*” THEN *buys_computer* = “*yes*”

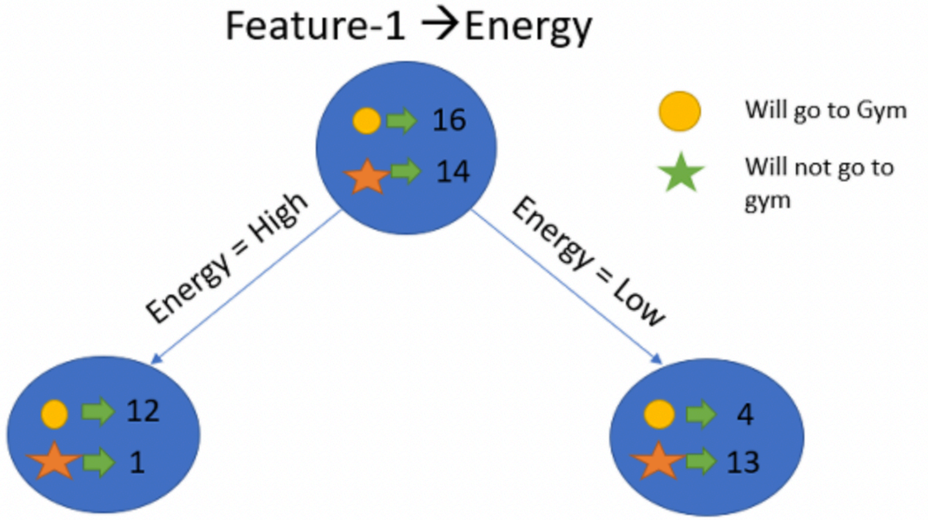
IF *age* = “<=30” AND *credit_rating* = “*fair*” THEN *buys_computer* = “*no*”

Decision Trees

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

https://www.slideshare.net/RajendraAkerkar/decision-tree-9146388



$$E(Parent) = -\left(\frac{16}{30}\right)\log_2\left(\frac{16}{30}\right) - \left(\frac{14}{30}\right)\log_2\left(\frac{14}{30}\right) \approx 0.99$$

$$E(Parent|Energy = "high") = -\left(\frac{12}{13}\right)\log_2\left(\frac{12}{13}\right) - \left(\frac{1}{13}\right)\log_2\left(\frac{1}{13}\right) \approx 0.39$$

$$E(Parent|Energy = "low") = -\left(\frac{4}{17}\right)\log_2\left(\frac{4}{17}\right) - \left(\frac{13}{17}\right)\log_2\left(\frac{13}{17}\right) \approx 0.79$$

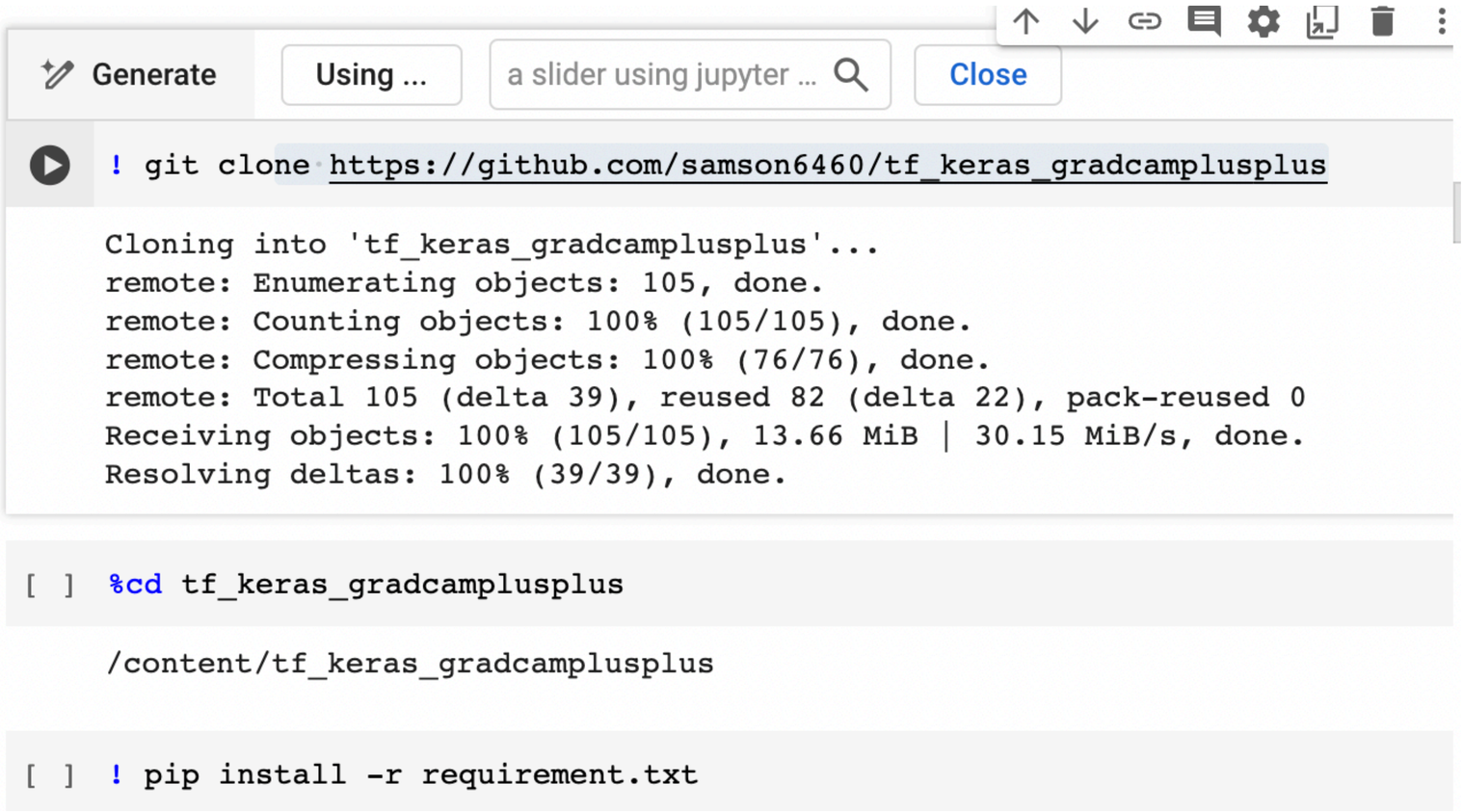
To see the weighted average of entropy of each node we will do as follows:

$$E(Parent|Energy) = \frac{13}{30} * 0.39 + \frac{17}{30} * 0.79 = 0.62$$

Now we have the value of E(Parent) and E(Parent|Energy), information gain will be:

$$\begin{aligned} Information\ Gain &= E(parent) - E(parent|energy) \\ &= 0.99 - 0.62 \\ &= 0.37 \end{aligned}$$

https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/


Generate
Using ...
a slider using jupyter ...
Close
! git clone https://github.com/samson6460/tf_keras_gradcamplusplus
Cloning into 'tf_keras_gradcamplusplus'...
remote: Enumerating objects: 105, done.
remote: Counting objects: 100% (105/105), done.
remote: Compressing objects: 100% (76/76), done.
remote: Total 105 (delta 39), reused 82 (delta 22), pack-reused 0
Receiving objects: 100% (105/105), 13.66 MiB | 30.15 MiB/s, done.
Resolving deltas: 100% (39/39), done.
[ ] %cd tf_keras_gradcamplusplus
/content/tf_keras_gradcamplusplus
[ ] ! pip install -r requirement.txt

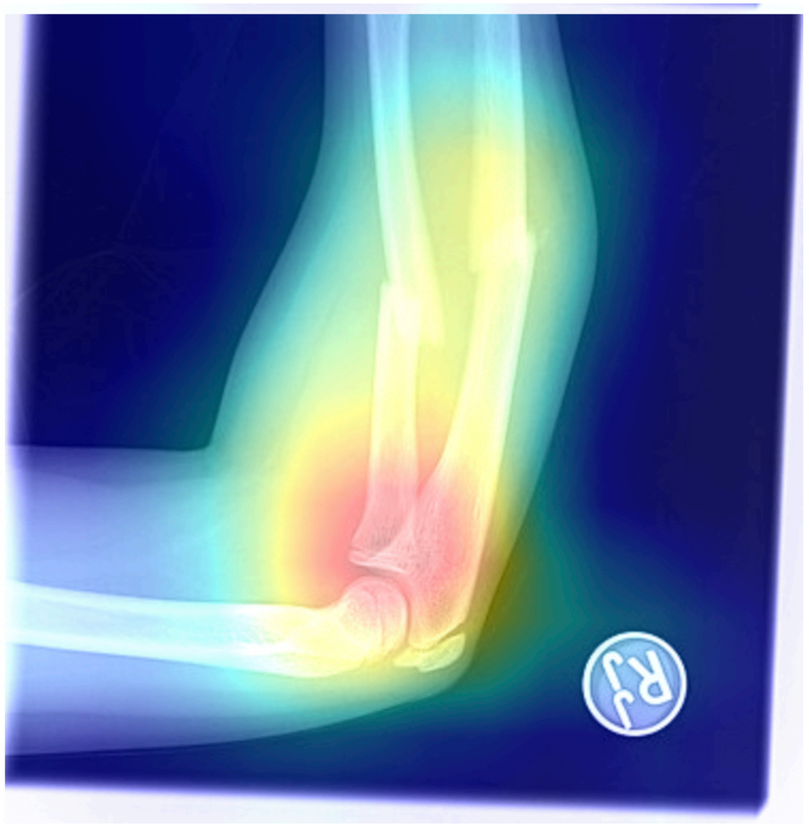
```
[ ]  from utils import vgg16_mura_model, preprocess_image, show_imgwithheat
     # from gradcam import grad_cam, grad_cam_plus

     # %% load the model
     model.summary()

     # %%
     # %% img_path = 'images/4320878114_30a836d428_z.jpg'
     # %% img = preprocess_image(img_path)

     # %% result of grad cam
     heatmap = grad_cam(model, test,
                        label_name = ['WRIST', 'ELBOW', 'SHOULDER'],
                        #category_id = 0,
                        )
     show_imgwithheat(img_path, heatmap)

     # %% result of grad cam++
     heatmap_plus = grad_cam_plus(model, img)
     show_imgwithheat(img_path, heatmap_plus)
```



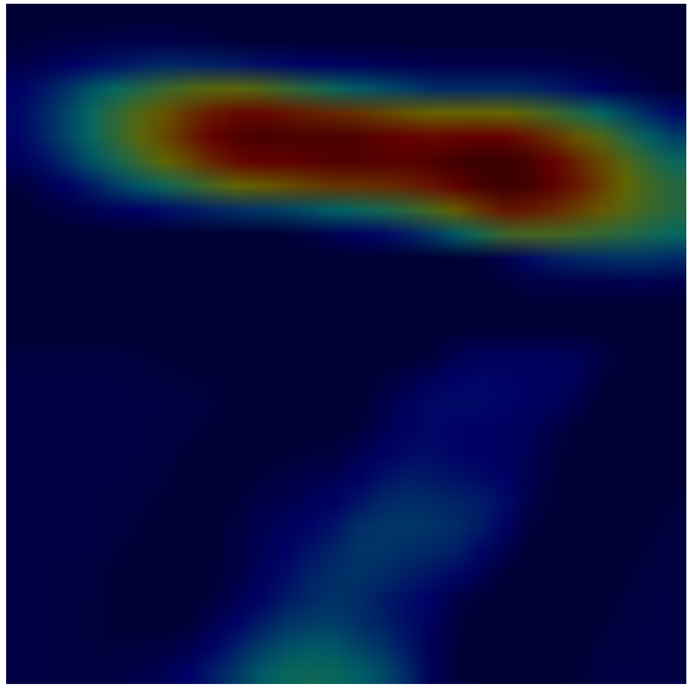
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 20, 20, 1)         82

 max_pooling2d_3 (MaxPoolin  (None, 10, 10, 1)         0
 g2D)

 flatten_3 (Flatten)         (None, 100)               0

 dense_3 (Dense)             (None, 10)                1010

=================================================================
Total params: 1092 (4.27 KB)
Trainable params: 1092 (4.27 KB)
Non-trainable params: 0 (0.00 Byte)
```



```
[ ]  show_imgwithheat("test.png", heatmap)

     # %% result of grad cam++
     heatmap_plus = grad_cam_plus(model, test.transpose([1,2,0]))
     show_imgwithheat("test.png", heatmap_plus)
```

7

GRAD-CAM

- **Gradient-weighted Class Activation Mapping (Grad-CAM)** generalizes CAM for a wide variety of CNN-based architectures
  - i.e., without requiring architectural changes or re-training

- Characteristics
  - Without GAP layer, we need a way to define weights – w_k^c
  - → Grad-CAM uses the gradients of any target concept (c) (e.g., 'dog' in a classification network) flowing into the final convolutional layer, and derive summary statistics out of it to represent the weights (importance)

GRAD-CAM

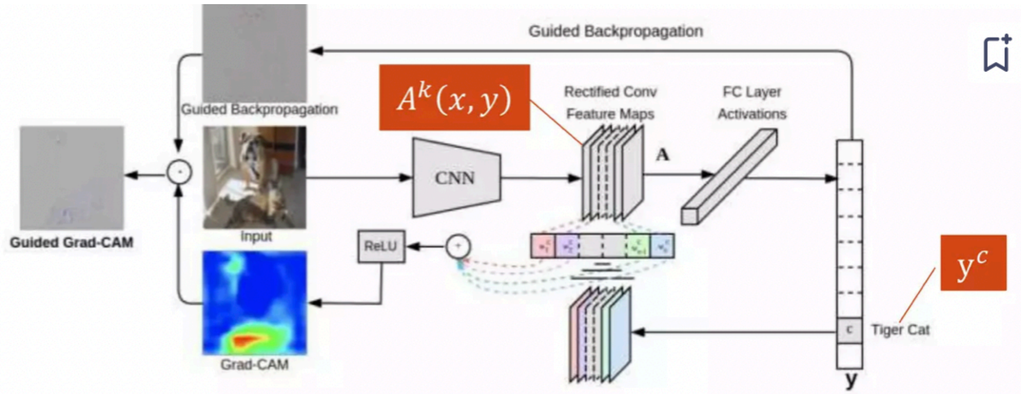


Figure 1: Grad-CAM overview: Given an image, and a category ('tiger cat') as input, we foward propagate the image through the model to obtain the raw class scores before softmax. The gradients are set to zero for all classes except the desired class (tiger cat), which is set to 1. This signal is then backpropagated to the

- Procedure
  - For a given class c, compute the gradient of its score– y^c (before the softmax), w.r.t. each feature map activations $A^k \in \mathbb{R}^{u \times v}, k = 1, \dots, n$ of a convolutional layer, i.e. $\frac{\partial y^c}{\partial A^k} \in \mathbb{R}^{u \times v}$ ← Influence of $A^k(x,y)$ to y^c
  - Define the importance weights of feature map k via GAP:

$$\alpha_k^c = \frac{1}{Z} \sum_{i \in x} \sum_{j \in y} \frac{\partial y^c}{\partial A_{ij}^k}$$

GRAD-CAM

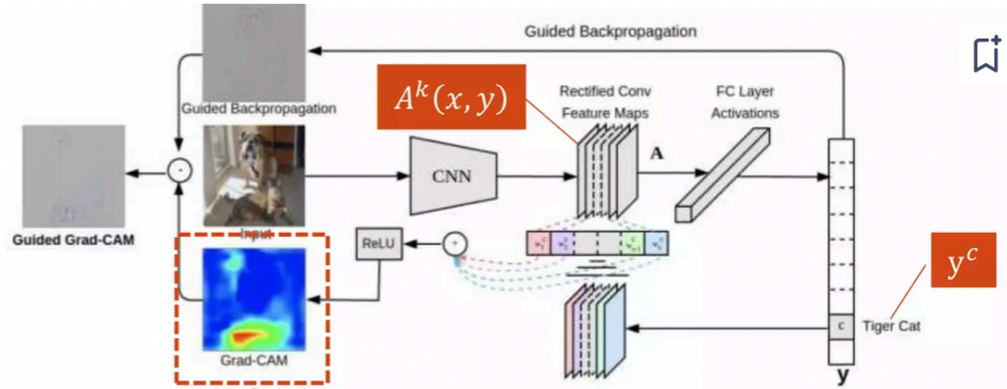


Figure 1: Grad-CAM overview: Given an image, and a category ('tiger cat') as input, we foward propagate the image through the model to obtain the raw class scores before softmax. The gradients are set to zero for all classes except the desired class (tiger cat), which is set to 1. This signal is then backpropagated to the

- Procedure
  - → Compute Grad-CAM:

$$L^c_{Grad-CAM}(x,y) = ReLU\left(\sum_k \alpha_k^c \; A^k(x,y)\right) \in \mathbb{R}^{u \times v}$$

  - ReLU is applied because we are only interested in the features (neurons) that have a positive influence on the class of interest
  - i.e. pixels whose intensity should be increased in order to increase y^c

Note: if the shape (u, v) of $L^c_{Grad-CAM}$ is different from that of input images, up-sampling is needed to equalize the