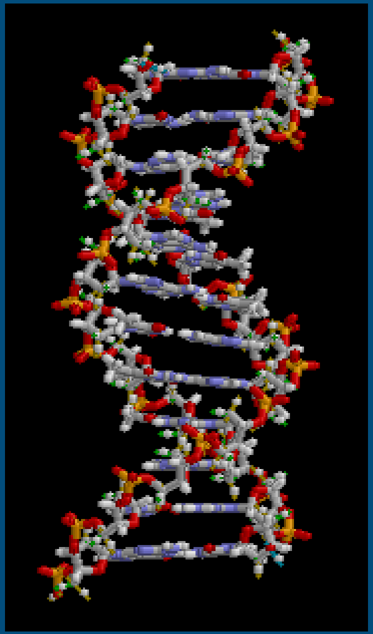
2018 Predictive Analytics Symposium

Session 14: AP - Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) - Moving Beyond Basic Neural Networks for Innovative Advantages



2019 PREDICTIVE ANALYTICS SYMPOSIUM

Session 14: AP - Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) - Moving Beyond Basic Neural Networks for Innovative Advantages

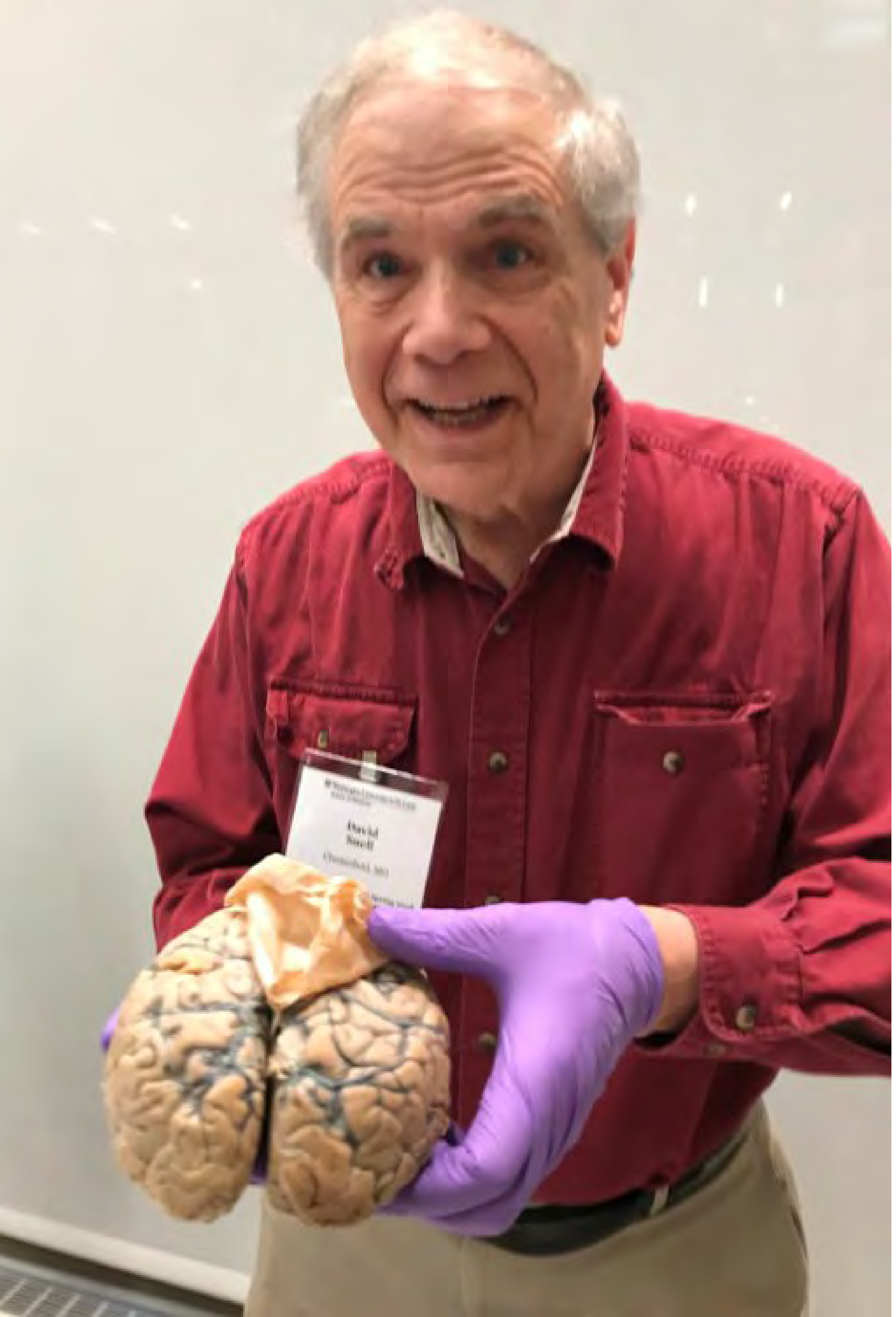
19-September-2019

David Snell, FALU, FLMI, ASA, MAAA, CLU, ChFC, ARA, ACS, MCP dave@ActuariesAndTechnology.com
Balathasan Giritharan Ph. D giritharanb@gmail.com

SOCIETY OF ACTUARIES®

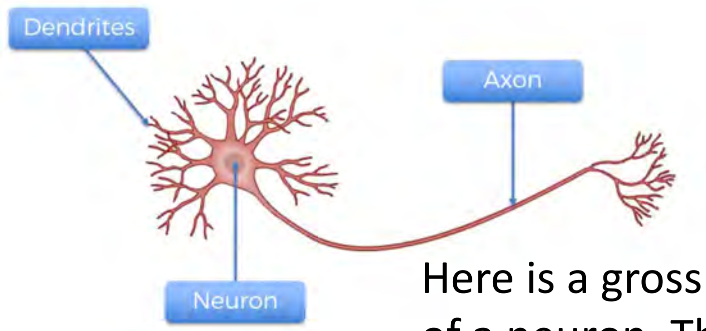
Overview of this session

- Neural Networks
- Artificial Neural Networks
- Deep Neural Networks
- Digitization of pictures
- Convolution and Pooling
- Convolutional Neural Networks
- Generative Adversarial Networks
- Applications
- Q & A
- Glossary of common terms (for later reference)



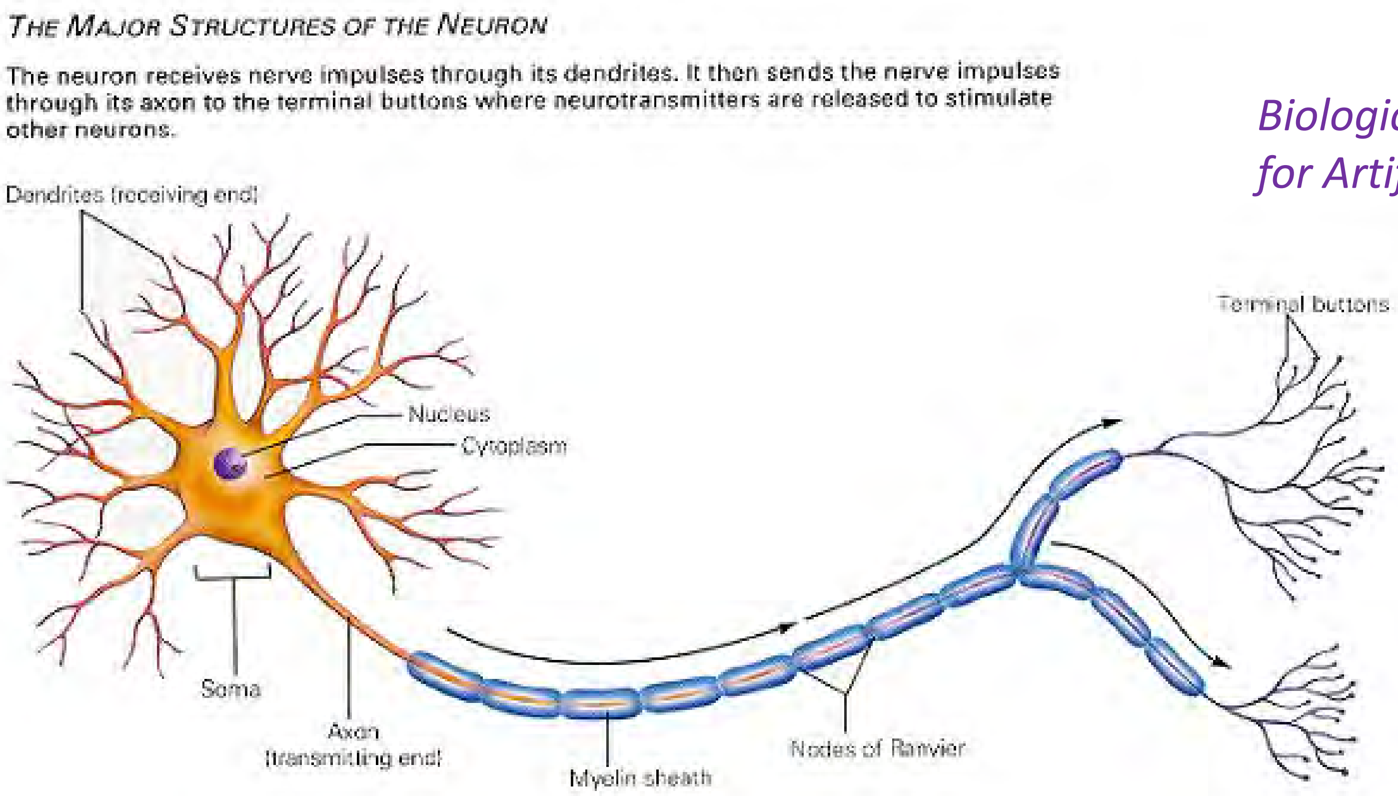
Real Neural Networks

The flap you see on the real human brain in this picture is the meninges, which is a protective covering between the skull and the brain. When infected, the resulting disease is called meningitis.



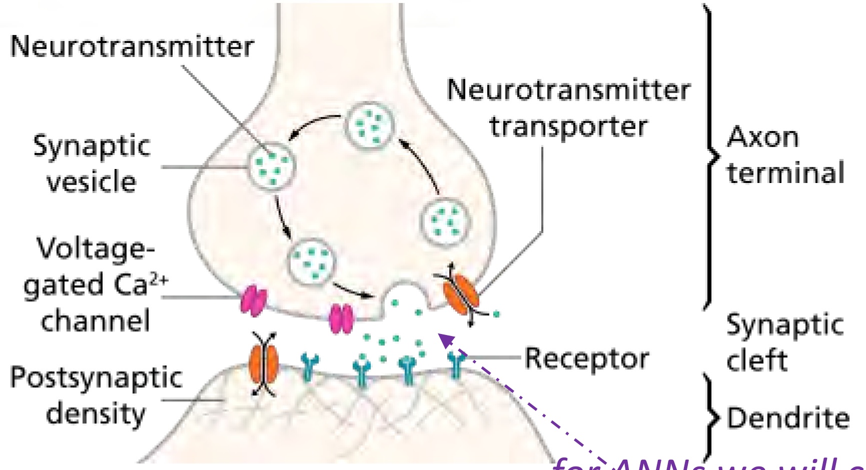
Here is a grossly oversimplified depiction of a neuron. The human brain has about 86 Billion neurons.

The brain I am holding in this picture has been injected with dye to highlight blood vessels.



Biological Neuron (still oversimplified)
*for Artificial Neural Networks, we will call this a **node***

I have added some slides to the end of this deck with definitions of some key terms



*for ANNs we will call the synapse (aka, channel, gap, connector) an **edge***

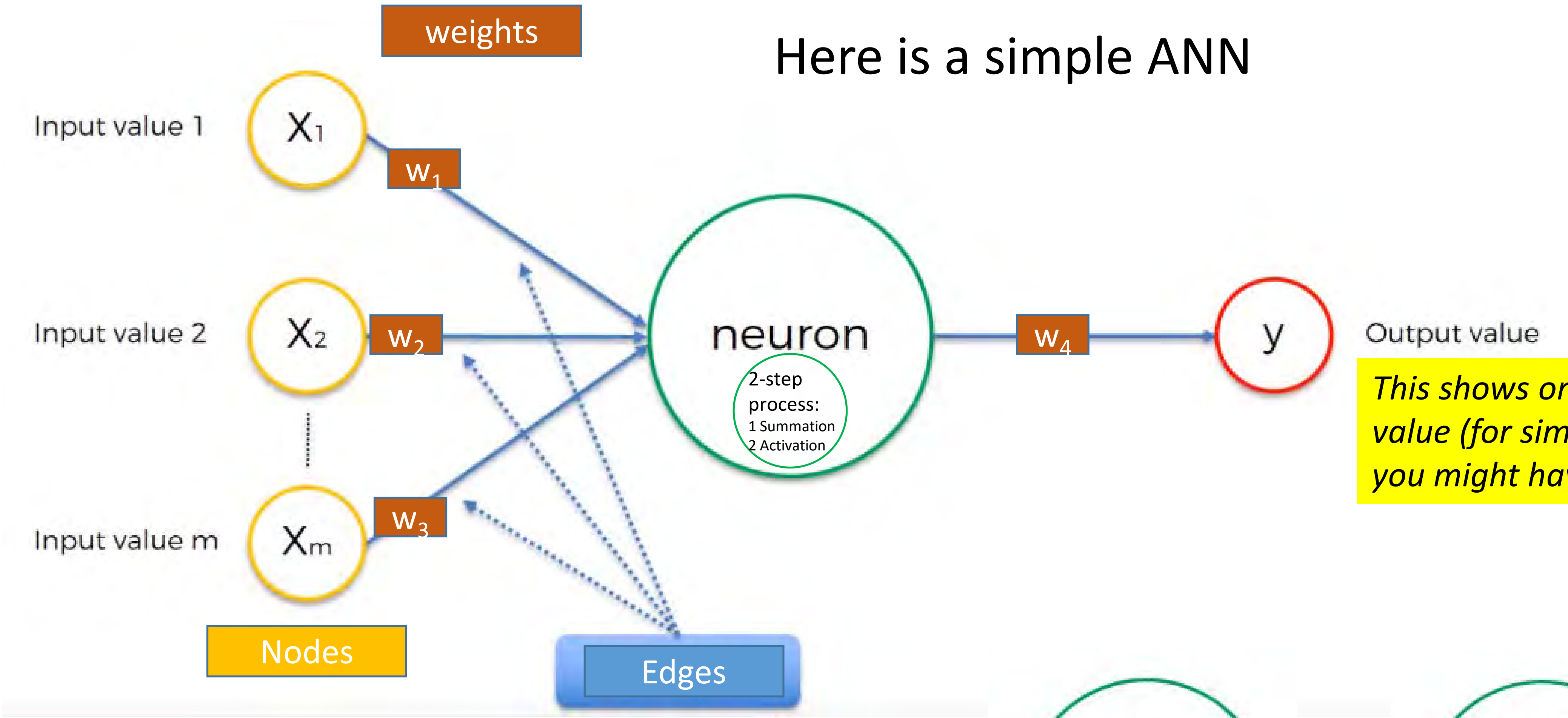
We have 100s of trillions of synapses ➔

*for ANNs we will call the mix of Ca, Na, soup and spark stuff a **weight***

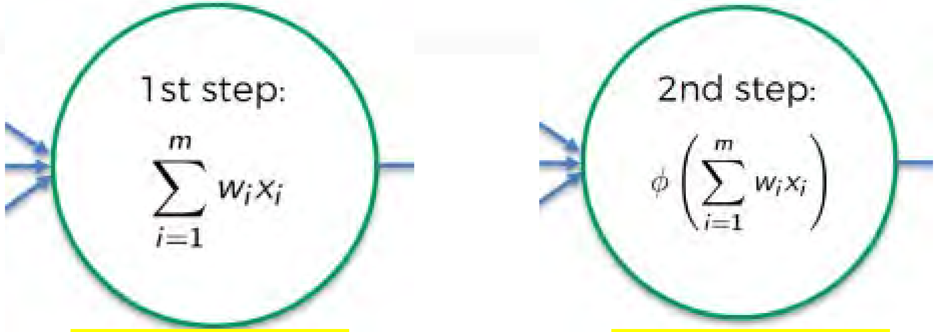
What is an Artificial Neural Network?

- In the context of machine learning (as opposed to biology), a neural network (NN) is a type of machine learning algorithm.
- It should properly be termed an artificial neural network (ANN) since it is nature-inspired, but it does not occur in nature. We create these algorithms (directly, or indirectly).
- Like the neural network of a brain, ANNs learn by example.
- Unlike the neural network of a brain, ANNs are not (currently) general purpose solution algorithms. They are built to solve a specific type of task.

Here is a simple ANN

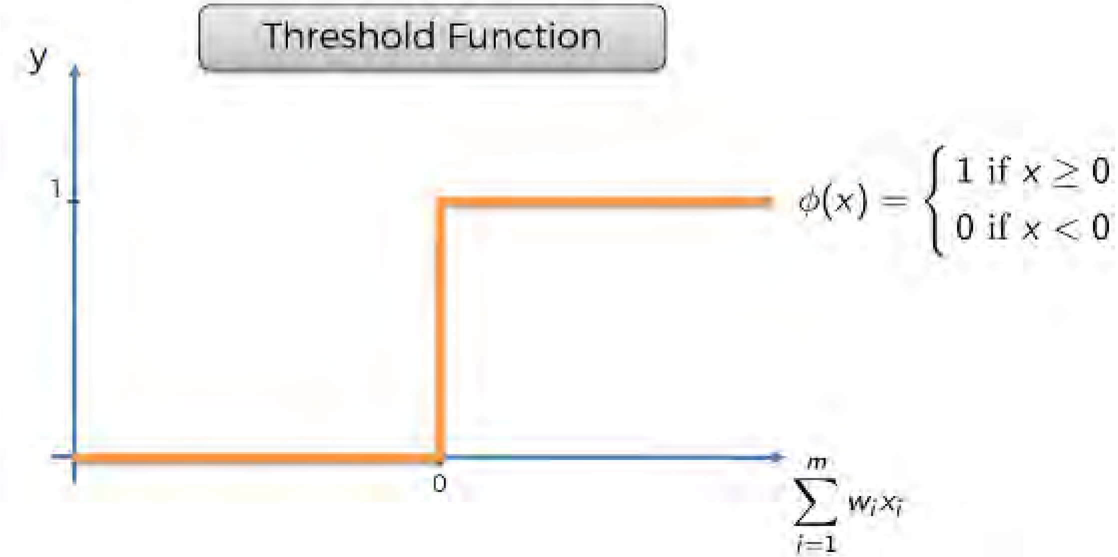


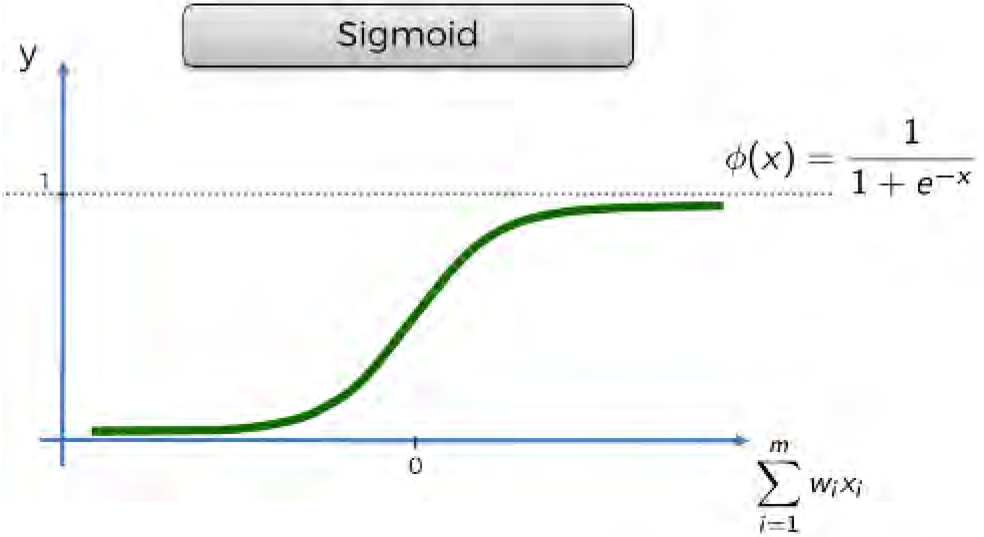
This shows one output value (for simplicity) but you might have several.



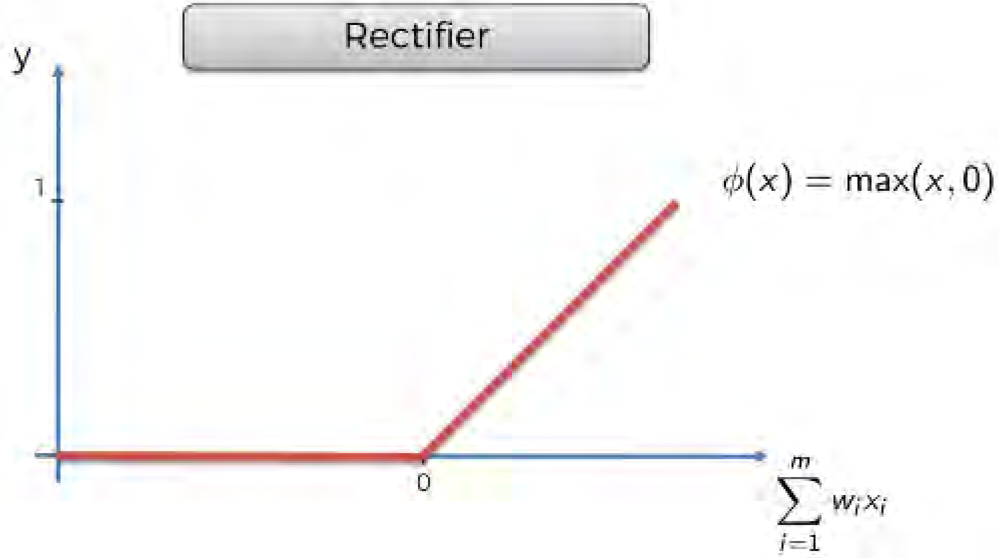
Summation

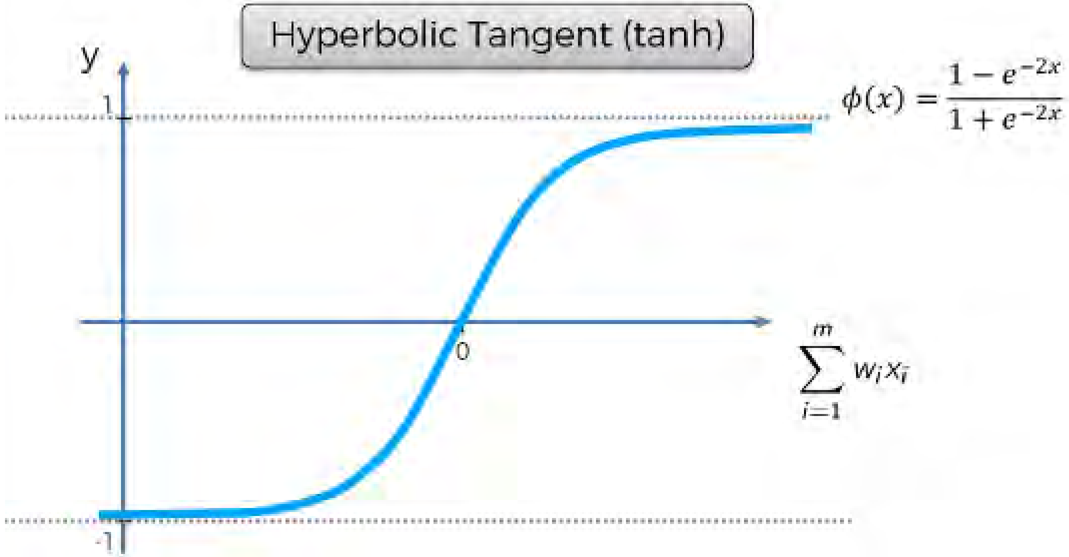
Activation



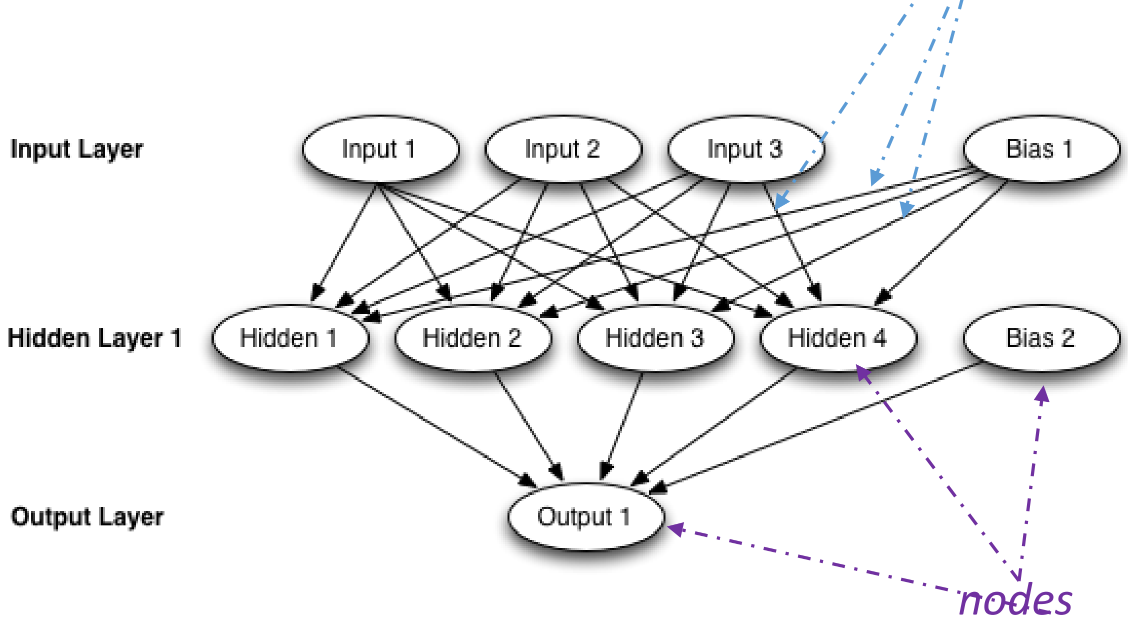


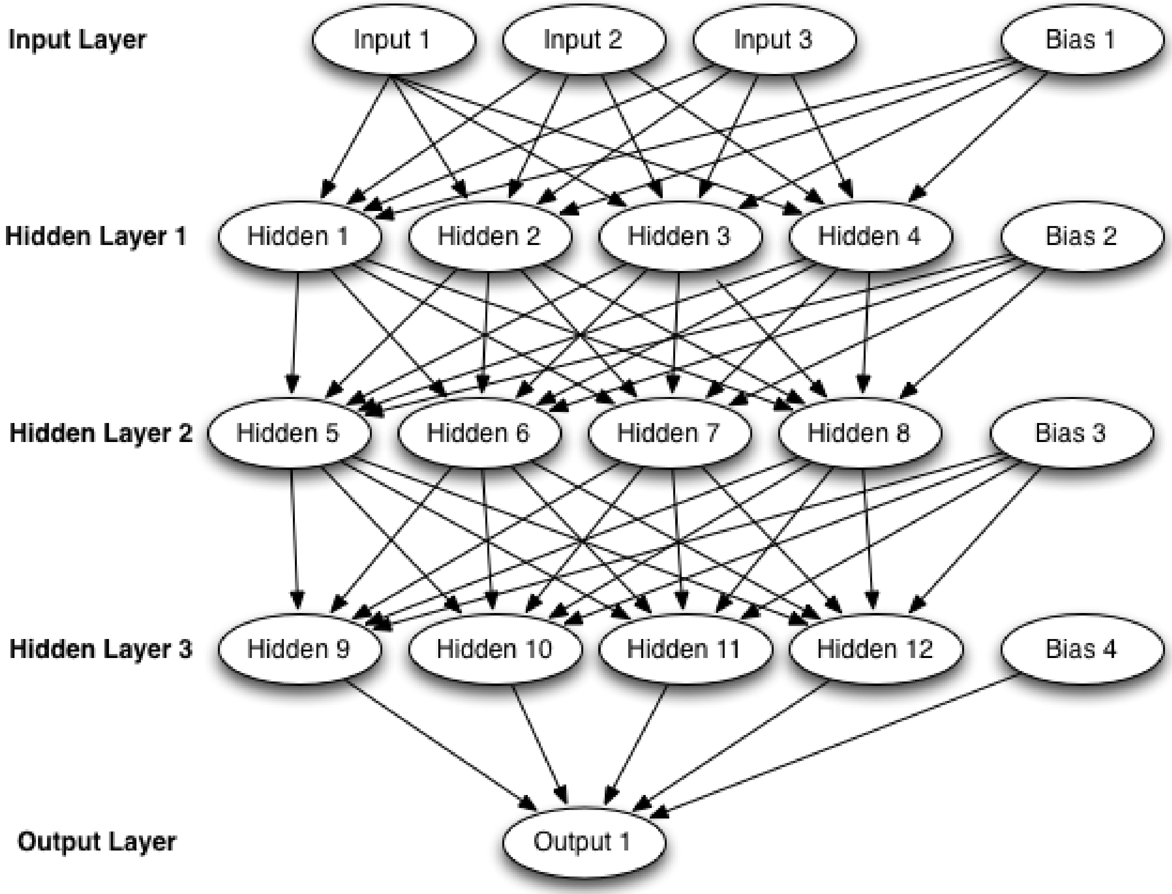
An activation function can take many forms.





- Artificial Neural Networks

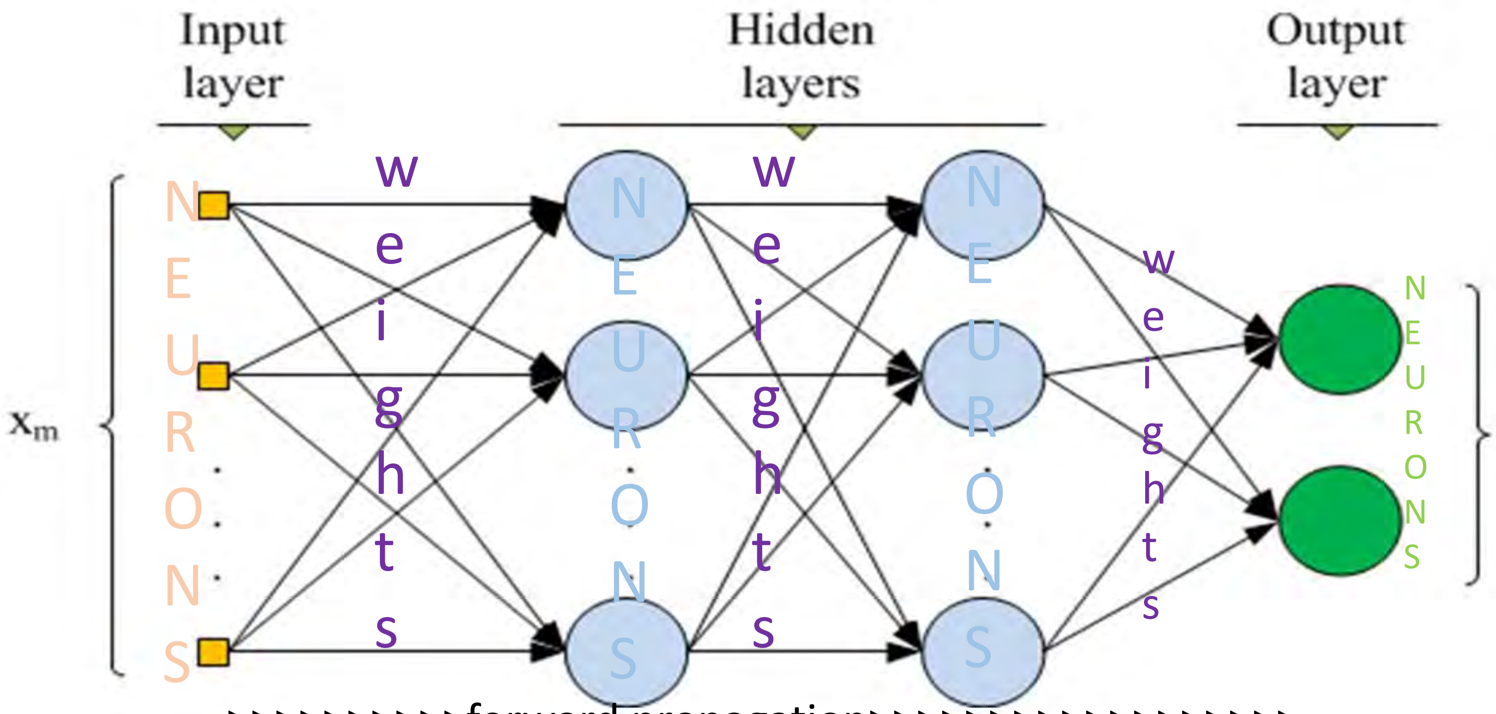




An ANN generally has three or more layers:

- Input layer
- Hidden layer(s)
- Output layer

Images used with permission from: Heaton, J. (2015). *Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks*. St. Louis, MO: Heaton Research, Inc, 1505714346.



$\hat{Y}_m$ (output values)

Y_m = actual values

>>>>>>>>>>>forward propagation>>>>>>>>>>>>>>>>>>>>

<<<<<back propagation<<<<<<<<<<<<<<<<<<<<<

Cost Function, $C = ½ \text{Sum}[(Y – \hat{Y})^2]$

Randomly initialize weights to small numbers (but not zero)

Forward propagate vector of first observation to get predicted result $\hat{Y}$. Each feature goes into one node of input layer.

Compute and minimize cost function, then update weights according to how much they are responsible for errors (and multiplying by the learning rate). This is called back propagation.

Continue to update weights after each observation (reinforcement learning) or after a batch of observations (batch learning)

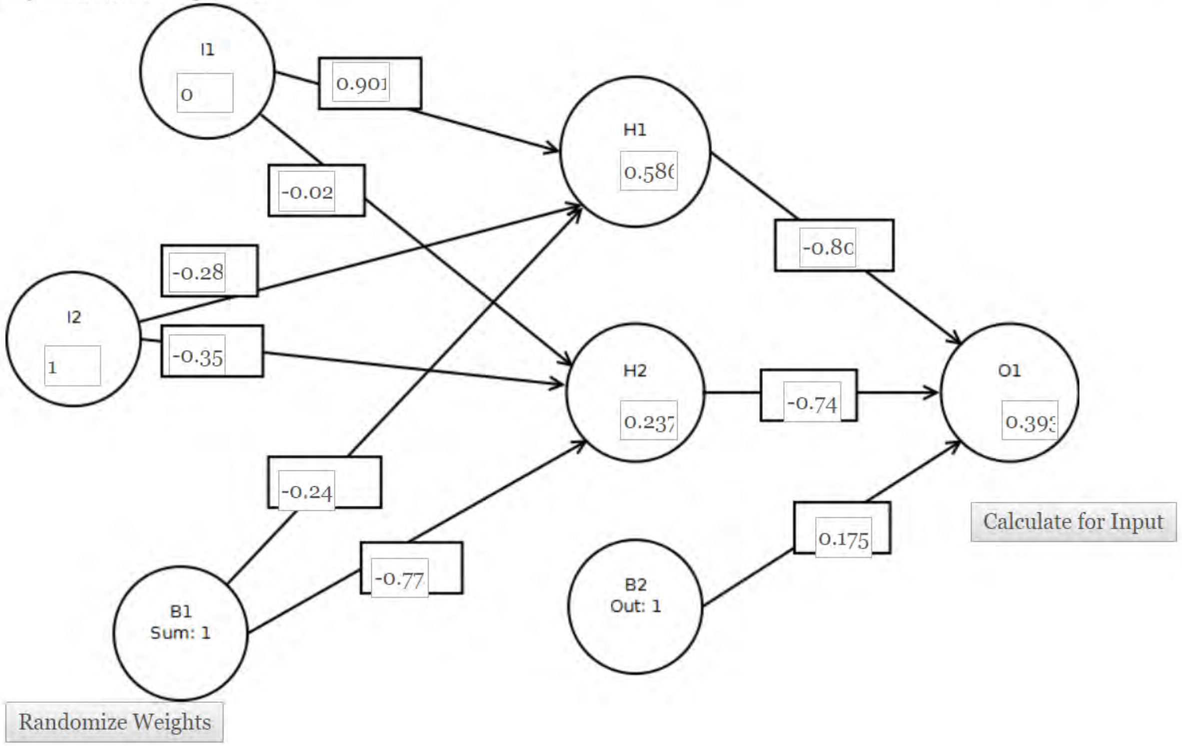
An epoch occurs when all of the training set has passed through the process. Redo for as many epochs as desired.

https://www.heatonresearch.com/aifh/vol3/xor_online.html

This example allows you to train a neural network using online backpropagation training. The training data used the XOR function, so the neural network should output the following:

```
When I1=0 and I2=0 then output 0
When I1=1 and I2=0 then output 1
When I1=0 and I2=1 then output 1
When I1=1 and I2=1 then output 0
```

Because this is online training you can train any of the above four patterns individually, or you can train all 4 at once. Either way, the weights are adjusted after each pattern.



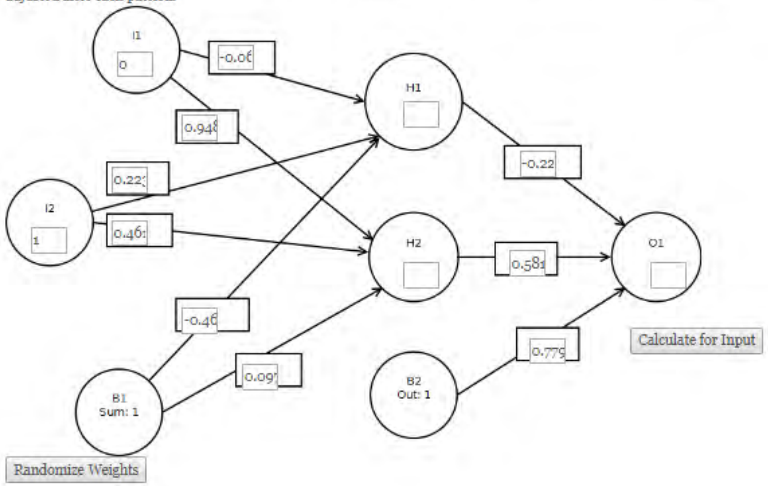
Artificial Intelligence for Humans

AIFH Volume 3, Chapter 6: XOR Online Training

This example allows you to train a neural network using online backpropagation training. The training data used the XOR function, so the neural network should output the following:

```
When I1=0 and I2=0 then output 0
When I1=1 and I2=0 then output 1
When I1=0 and I2=1 then output 1
When I1=1 and I2=1 then output 0
```

Because this is online training you can train any of the above four patterns individually, or you can train all 4 at once. Either way, the weights are adjusted after each pattern.



Training

Learning rate: 0.7, Momentum: 0.3

Mean Square Error(MSE):

- Input: [0,0], Desired Output: [0], Train Online
- Input: [1,0], Desired Output: [1], Train Online
- Input: [0,1], Desired Output: [1], Train Online
- Input: [1,1], Desired Output: [0], Train Online

Train All (same as clicking all 4 above)

Calculations

More Info

The following formulas are used in the above calculations.
The sigmoid function is calculated with the following formula:

$$S(t) = \frac{1}{1+e^{-t}}.$$

The derivative of the sigmoid function:

$$S'(x) = S(x) * (1.0 - S(x))$$

Mean Square Error(MSE) is calculated with the following formula:

$$\mathrm{MSE} = \frac{\sum_{t=1}^{n}(\hat{y}_t - y)^2}{n}.$$

Output Layer Error is calculated with the following formula:

$$E = (a - i)$$

Node delta is calculated with the following formula:

$$\delta_i = \begin{cases} -Ef'_i & \text{, output nodes} \\ f'_i \sum_k w_{ki}\delta_k & \text{, interior nodes} \end{cases}$$

Gradient is calculated with the following formula:

$$\frac{\partial E}{\partial w_{(ik)}} = \delta_k \cdot o_i$$

http://www.heatonresearch.com/aifh/vol3/xor_online.html

The human brain has evolved very slowly over many years. We spent the overwhelming majority of our existence as hunter gatherers not sitting at a desk behind a computer.

Our brains are not wired to efficiently process numbers.

Study the following sequence of numbers for 30 seconds:

6, 2, 4, 10 ,20, 24, 22, 16, 7, 9, 19, 17, 13

SOCIETY OF ACTUARIES®

How many of the numbers can you remember? Don’t feel badly, humans are not naturally good at this.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

6, 2, 4, 10 ,20, 24, 22, 16, 7, 9, 19, 17, 13

In fact, now you can probably remember them in order.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

An ordinary deep neural network is very good at discerning patterns in numbers.

6, 2, 4, 10 ,20, 24, 22, 16, 7, 9, 19, 17, 13

The average human can remember about 7 chunks of information at a time +/- 2 (5 to 9)

Try this set of letters:

S, L, M, I, I, F, N, A, U

Creating groups can improve the learning process

AI ML IS

FUN

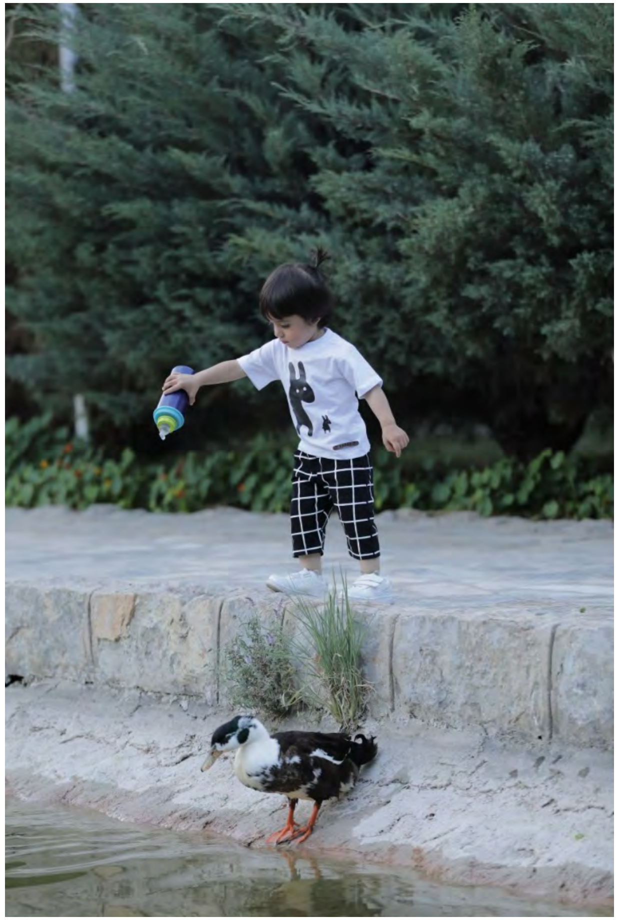
Since letters can easily be mapped to numbers (A➔1, B➔2, …) a neural network can also work with letters, words, and sentences for natural language processing (NLP)

(S, L, M, I, I, F, N, A, U)

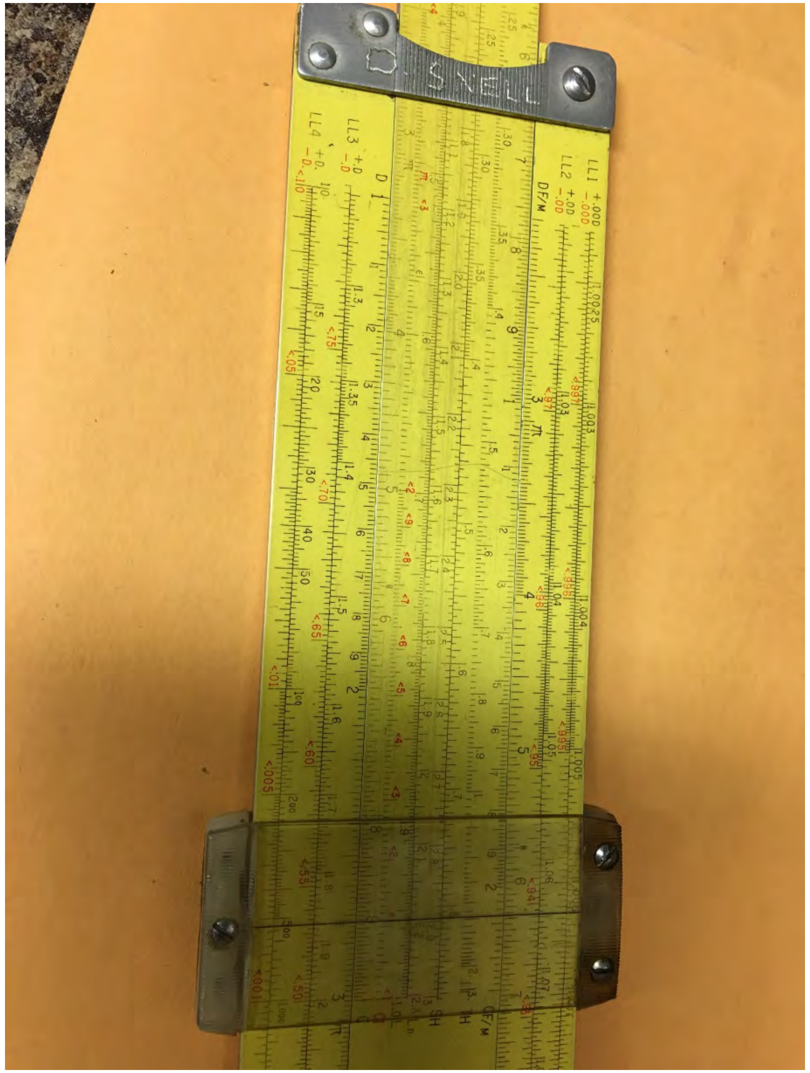
We have seen that humans are not good at remembering numbers, but very good at patterns.

Let’s see how just how good we are with those patterns.

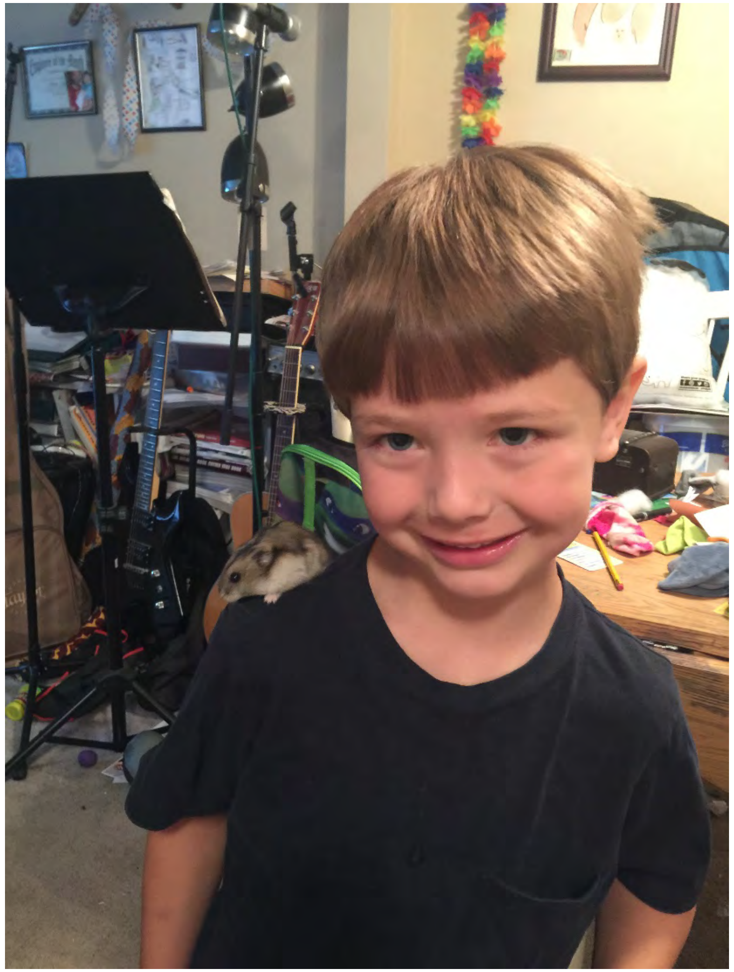
I will show you 20 slides for about 3 seconds apiece. After that one minute of study, we’ll have a test on them.

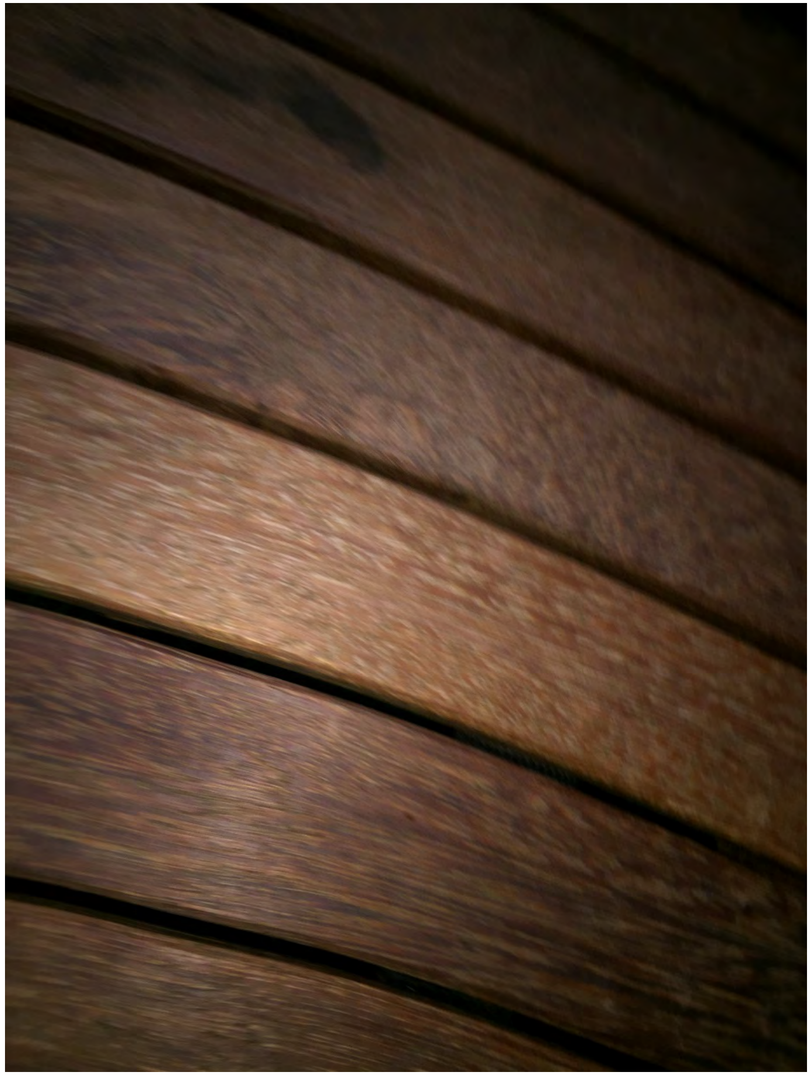



COSTCO
FOOD COURT
PIZZA
Fresh Baked
KIRKLAND




★ OFFICIAL ★
MAJOR LEAGUE BASEBALL






meineke
CAR CARE CENTER


IRON CITY GYM
300

Memory Test

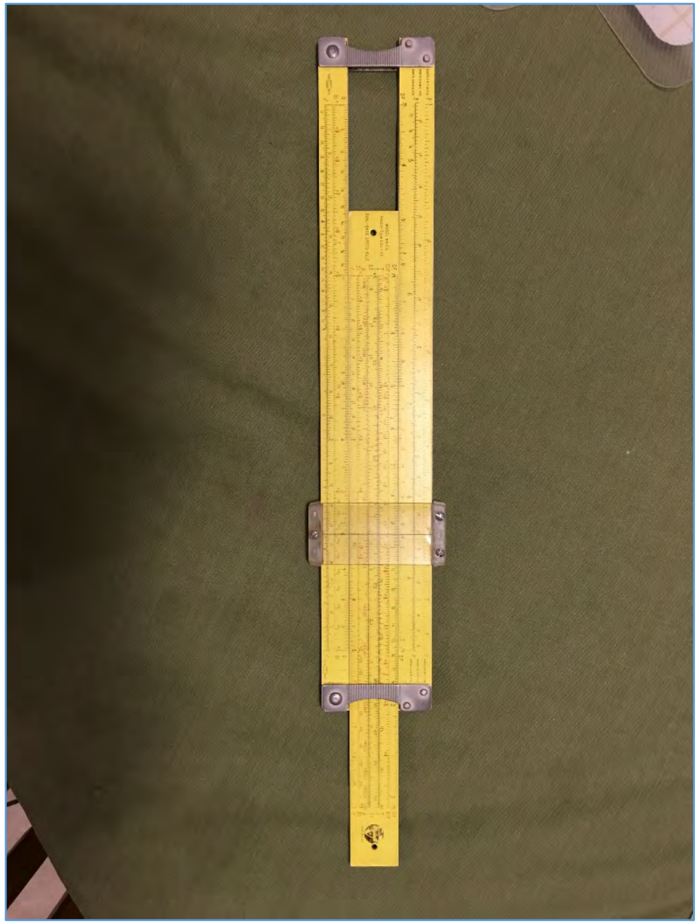
Now, you get to see another 20 slides for about 3 seconds each. Each slide will contain two pictures. One of the two will be from the 20 you saw already. The other one will be new.

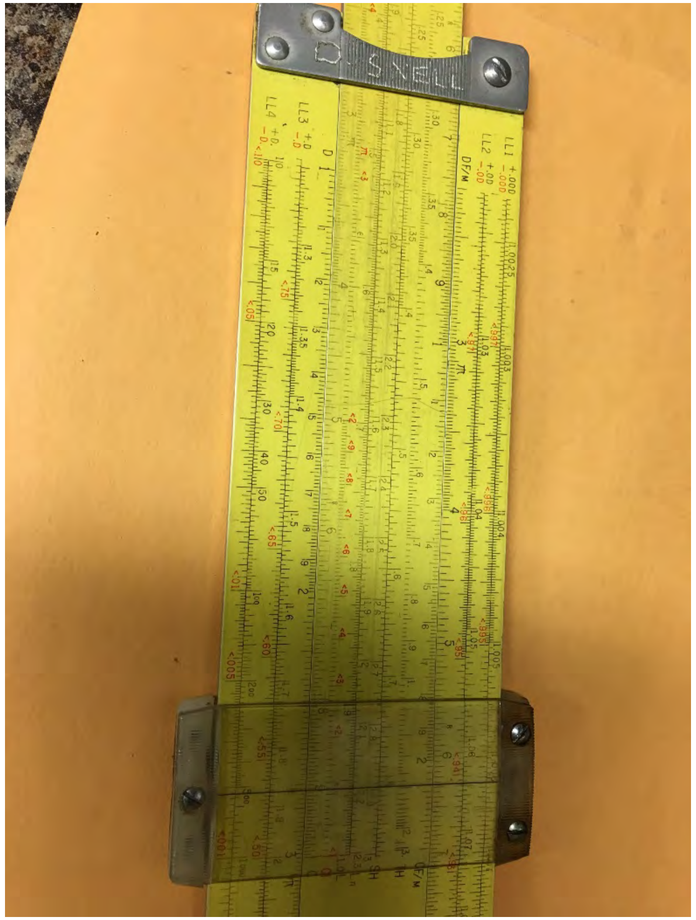
For each of the next 20 slides, please write **L** if the new slide is on the left, and **R** if the new slide is on the right.

This test was inspired by the Great Courses course ***Scientific Secrets for a Powerful Memory****, by Peter M. Vishton, Ph.D. Associate Professor of Psychology, College of William and Mary*

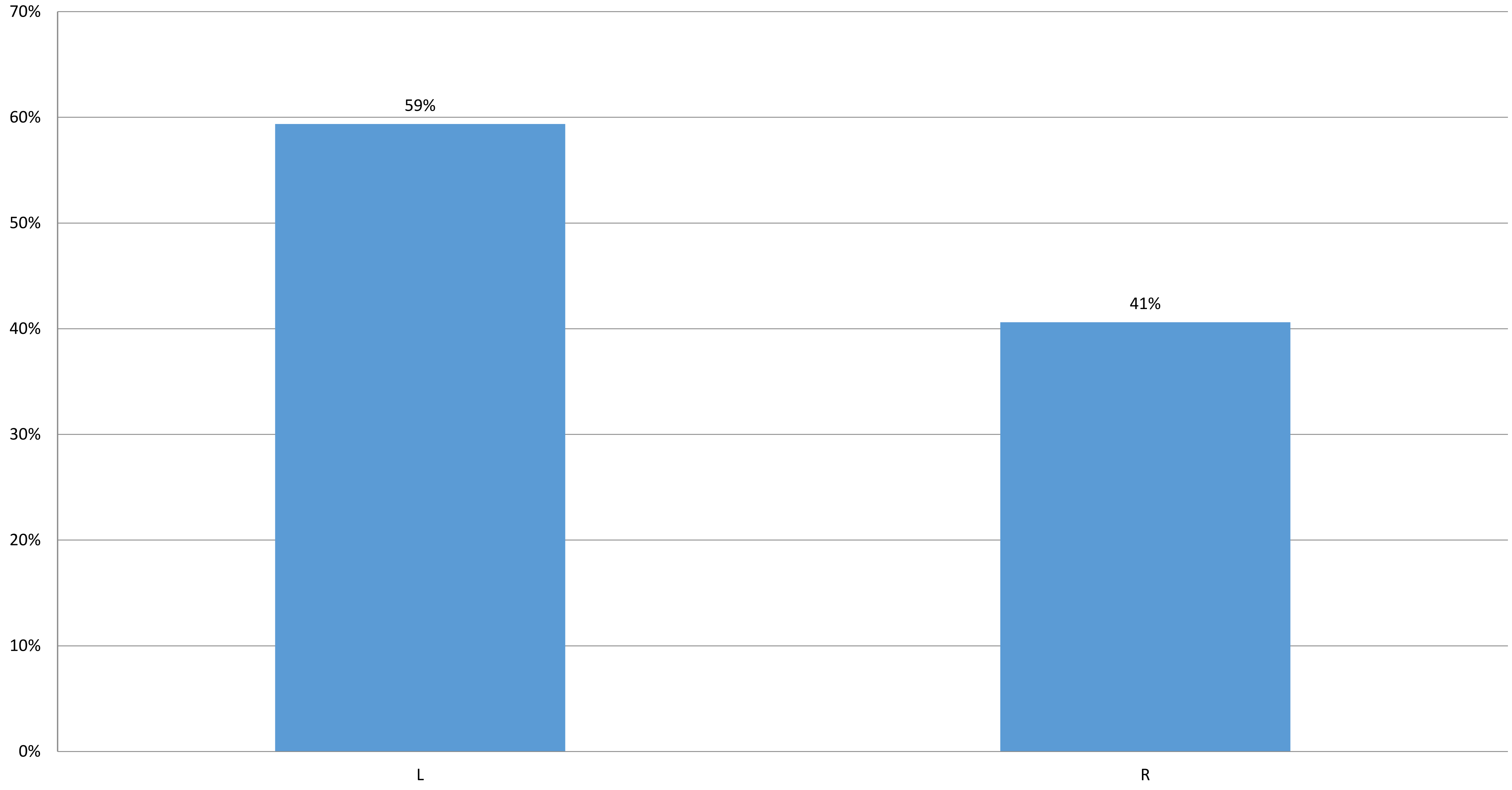
Mark **L** if the picture on the left is new, **R** if the picture on the right is new

1



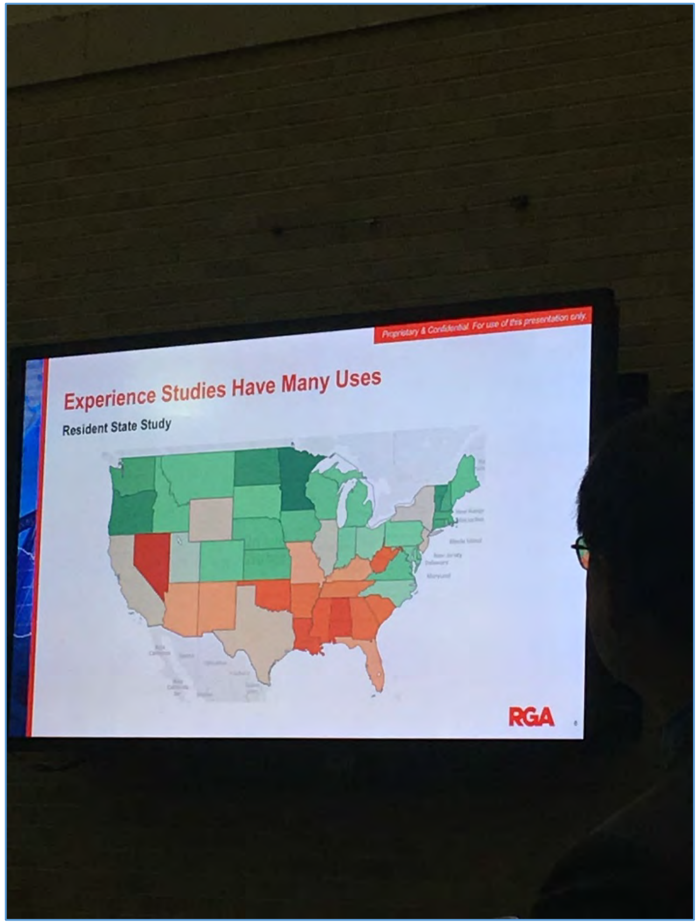


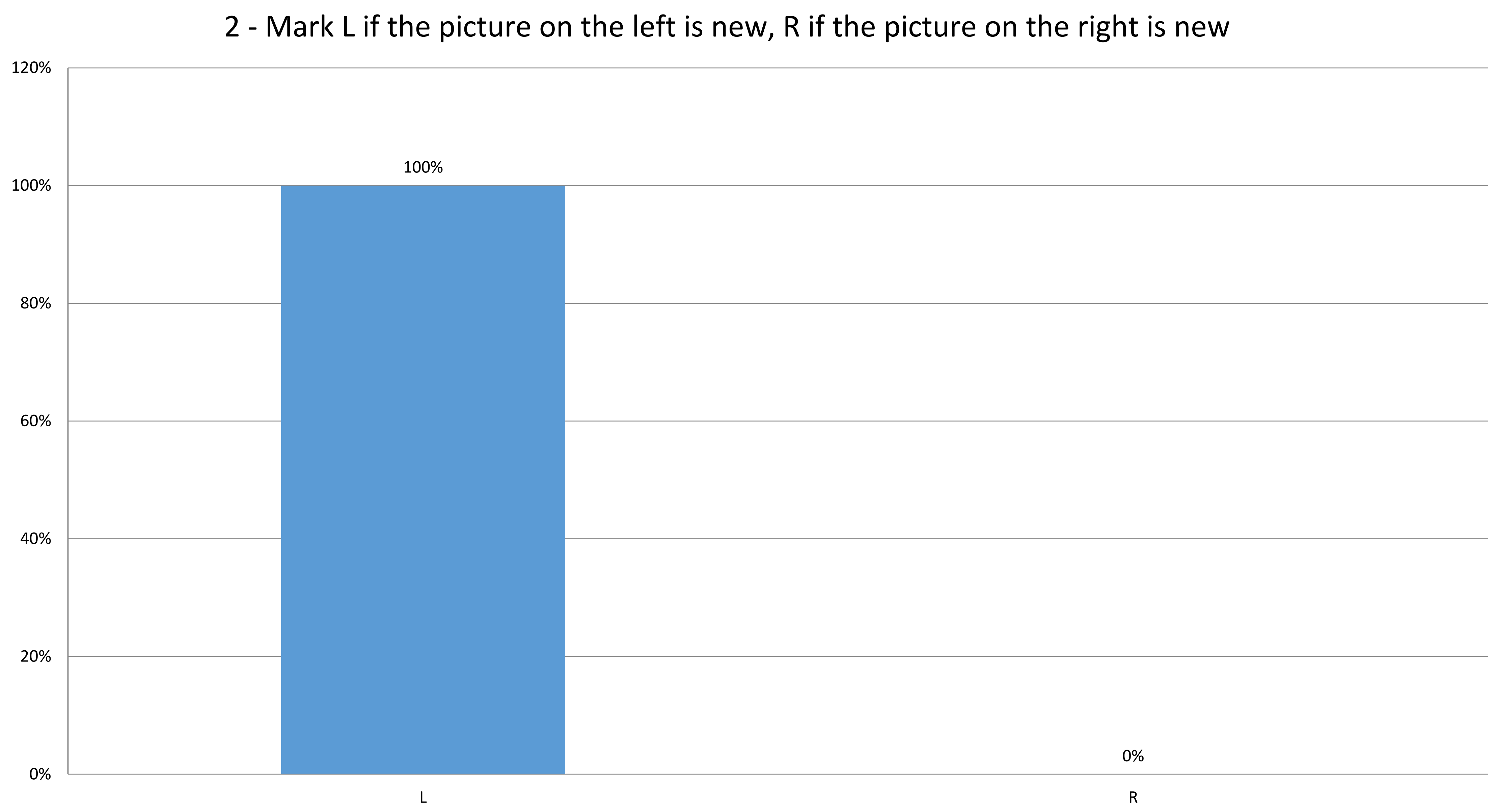
1 - Mark L if the picture on the left is new, R if the picture on the right is new



Mark L if the picture on the left is new, R if the picture on the right is new

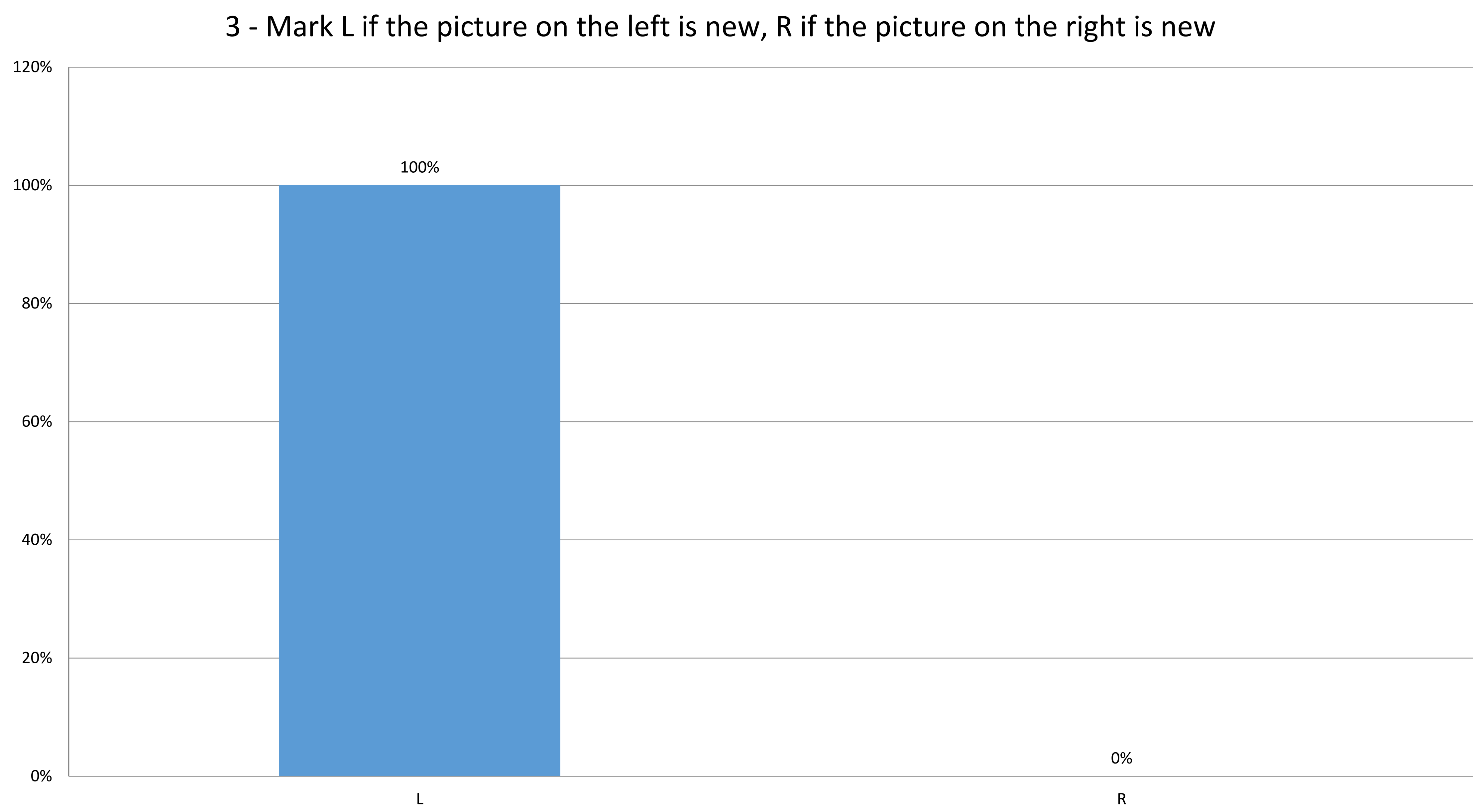
2




2 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

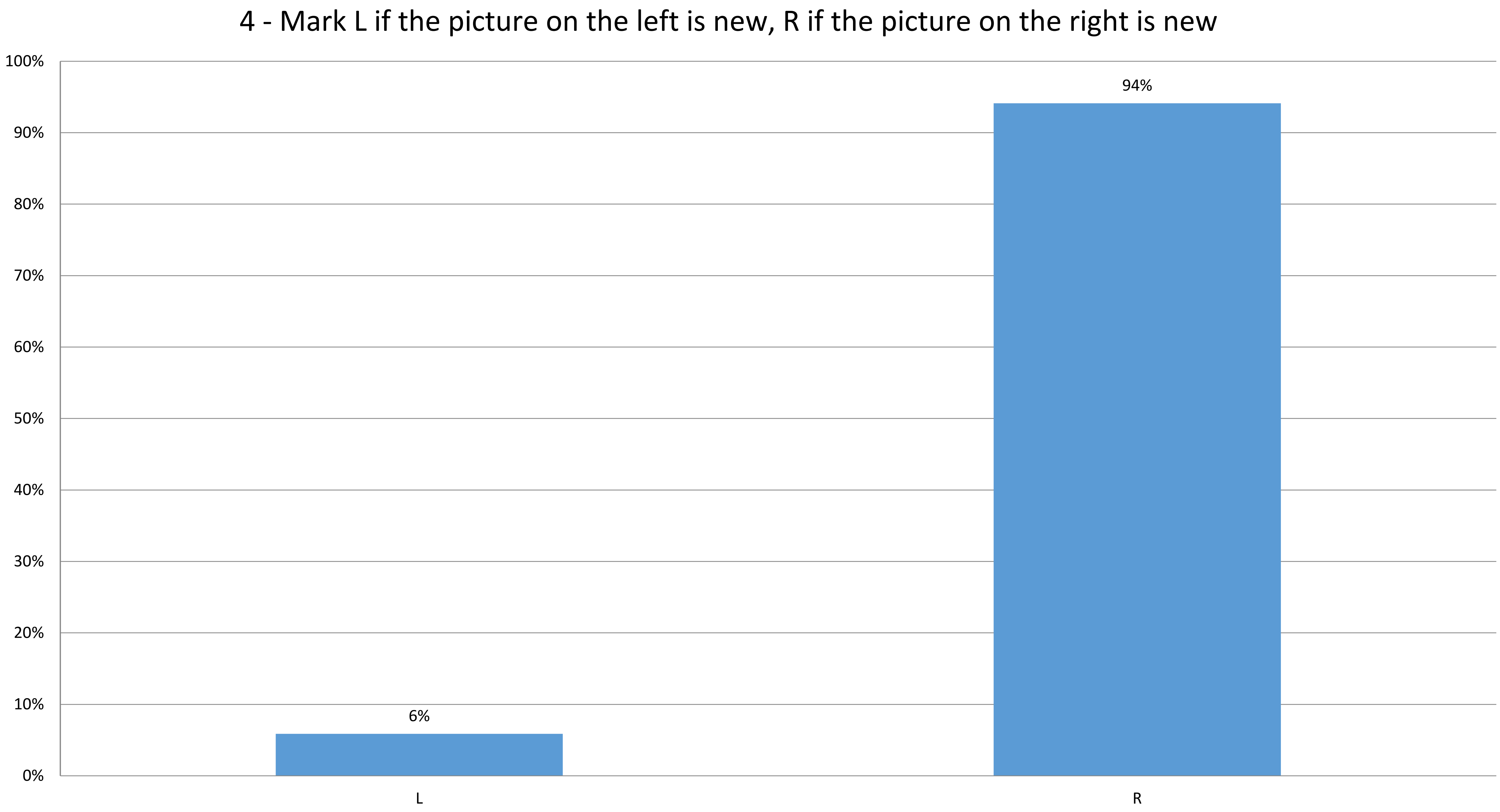
Mark L if the picture on the left is new, R if the picture on the right is new

3


3 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

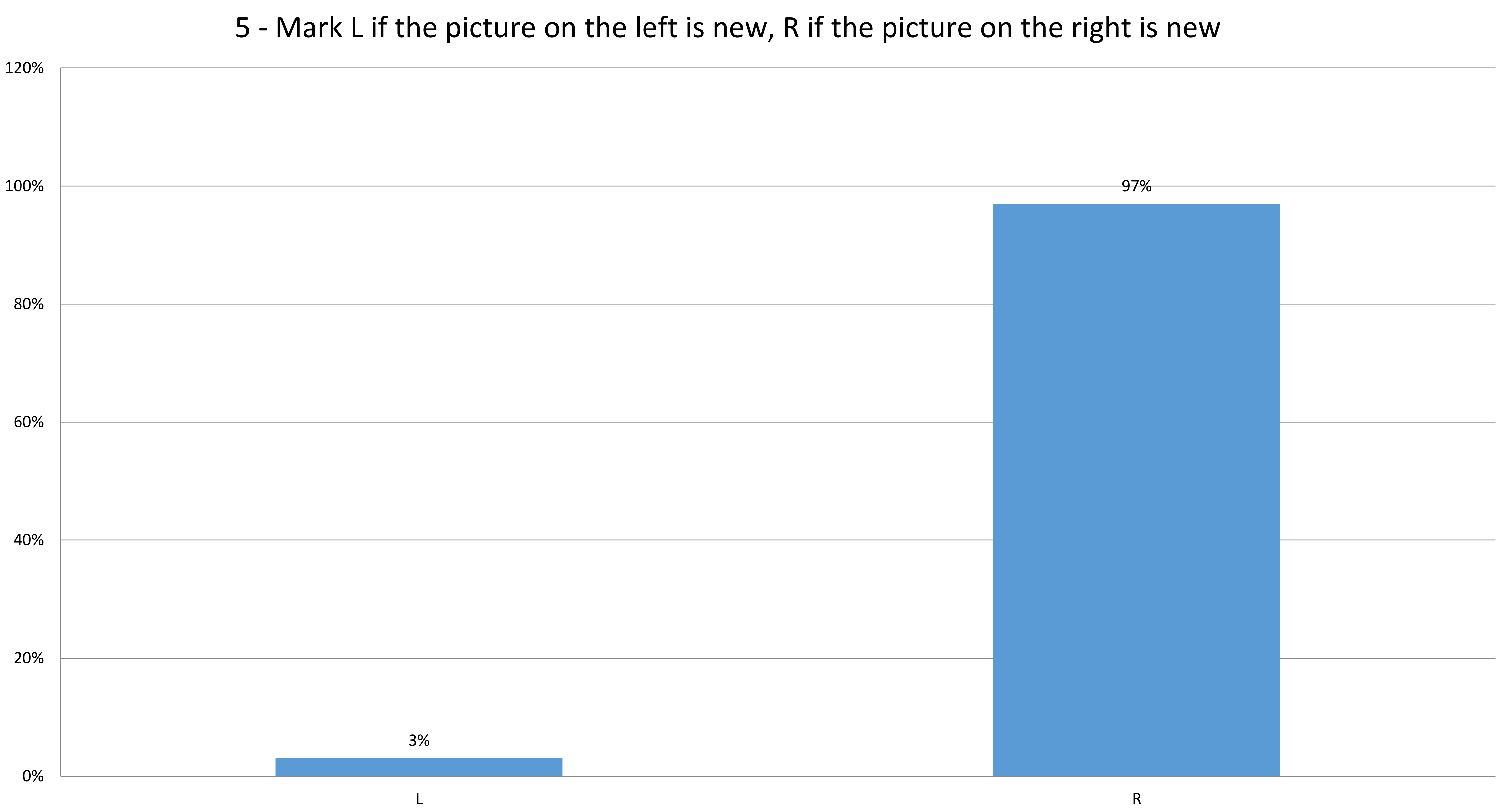
Mark L if the picture on the left is new, R if the picture on the right is new

4


4 - Mark L if the picture on the left is new, R if the picture on the right is new
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
94%
6%
L
R

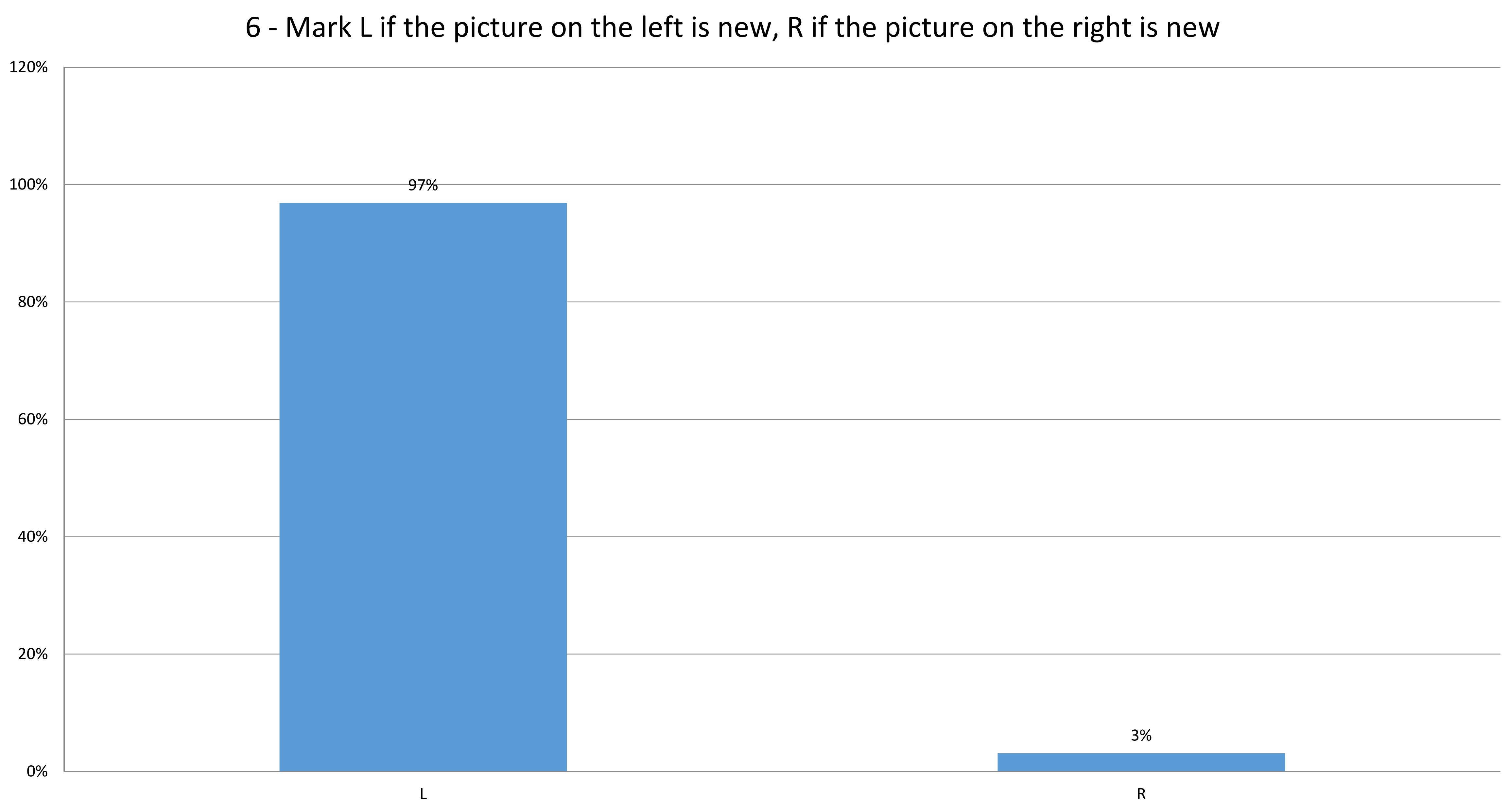
Mark L if the picture on the left is new, R if the picture on the right is new

5


5 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
3%
97%
L
R

Mark L if the picture on the left is new, R if the picture on the right is new

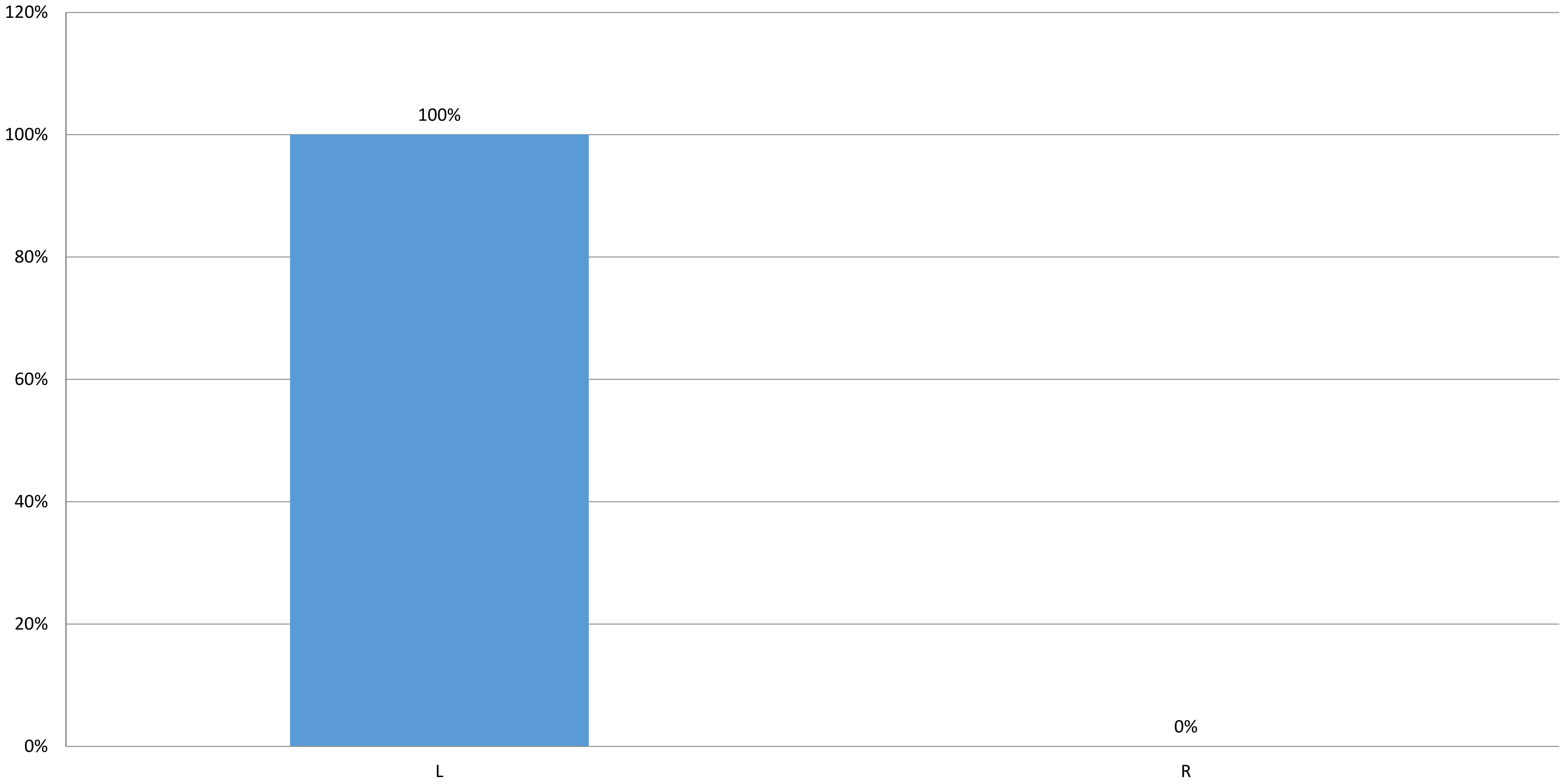
6


6 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
97%
3%
L
R

Mark L if the picture on the left is new, R if the picture on the right is new

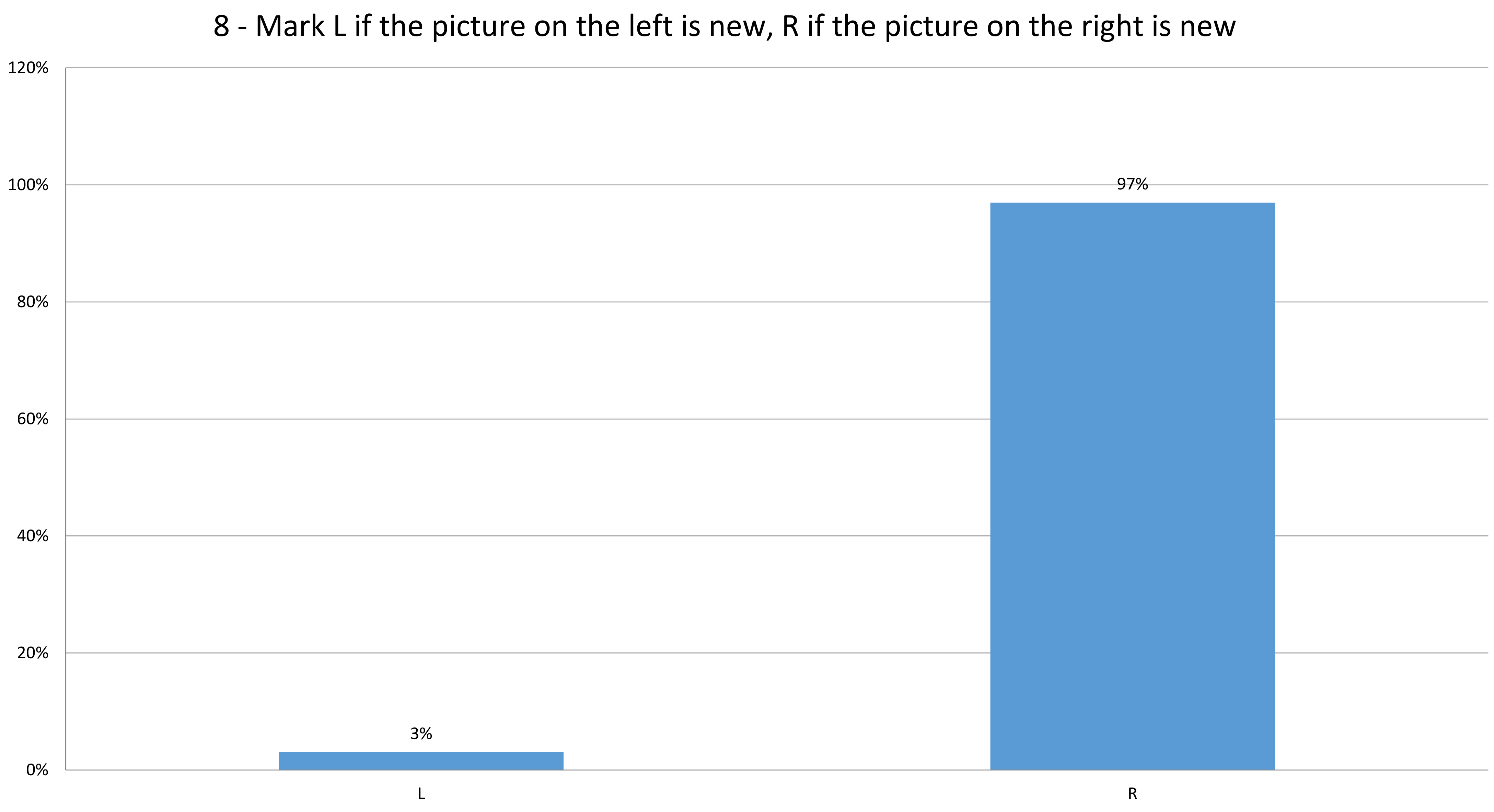
7

7 - Mark L if the picture on the left is new, R if the picture on the right is new



Mark L if the picture on the left is new, R if the picture on the right is new

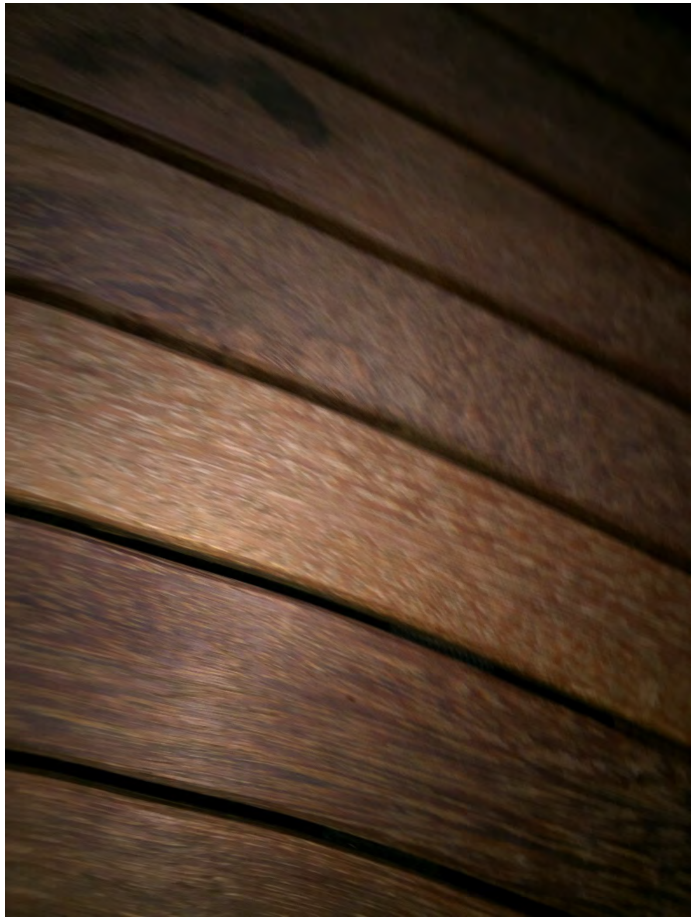
8

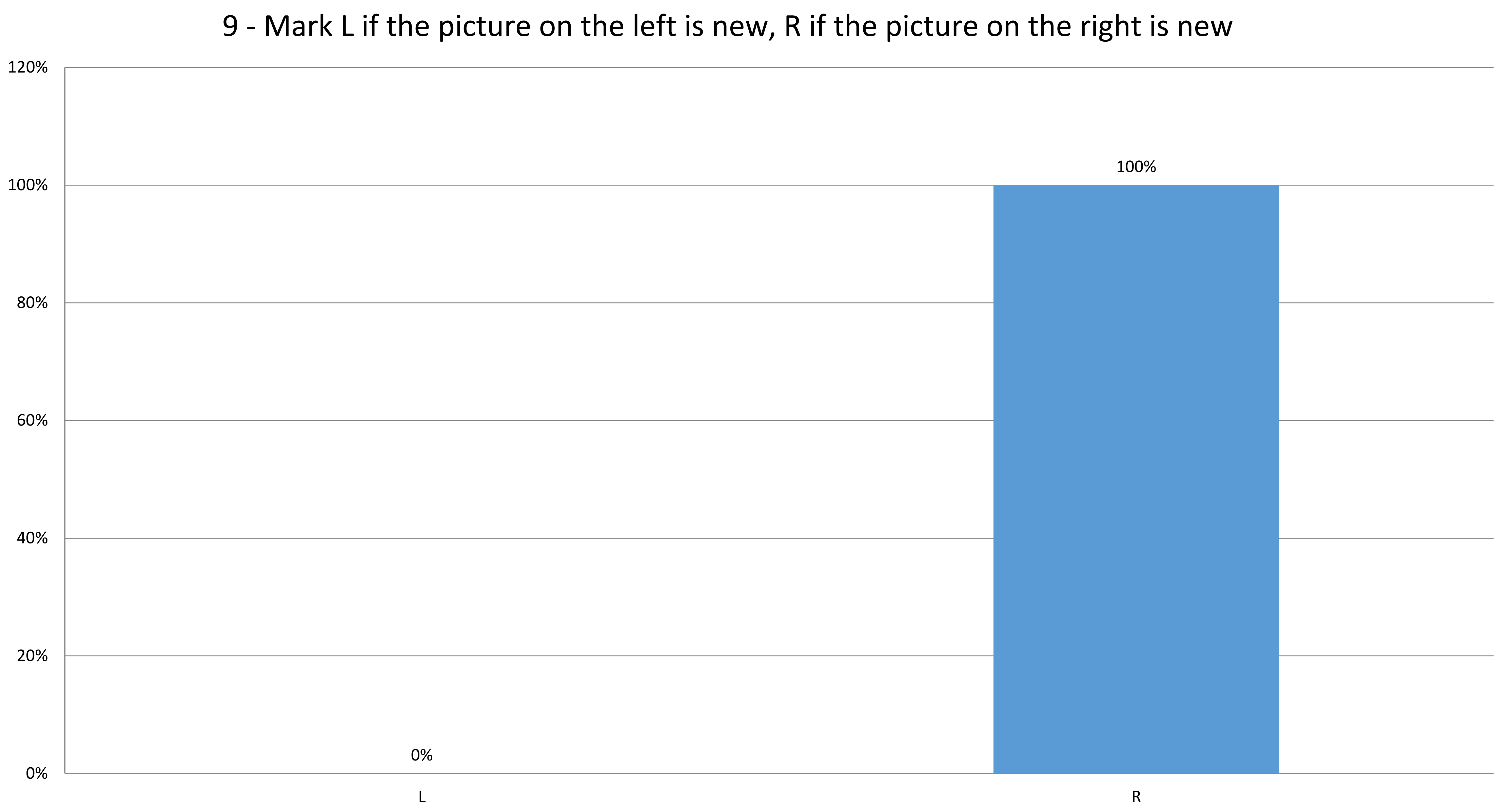


8 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
3%
97%
L
R

Mark L if the picture on the left is new, R if the picture on the right is new

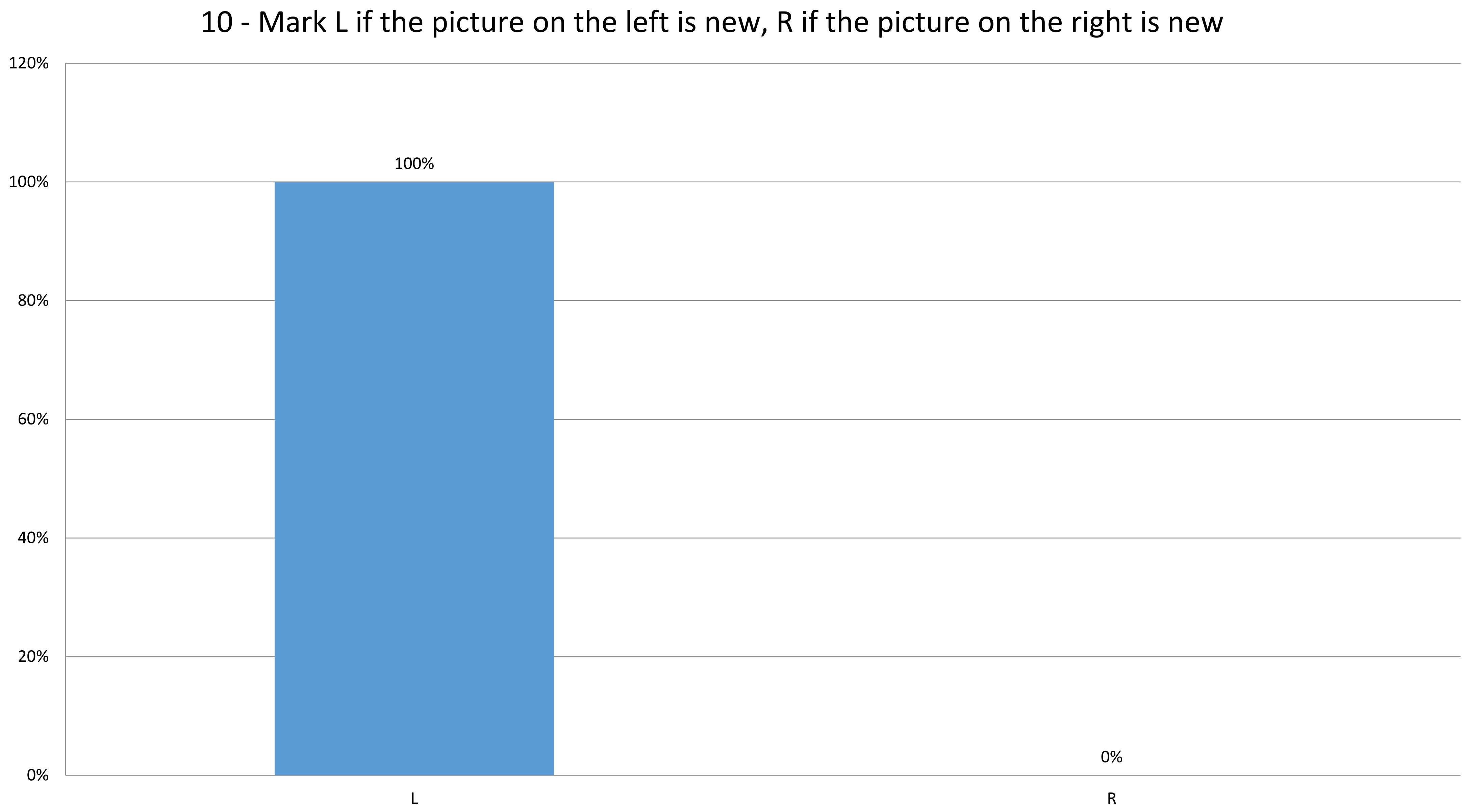
9




9 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
0%
100%
L
R

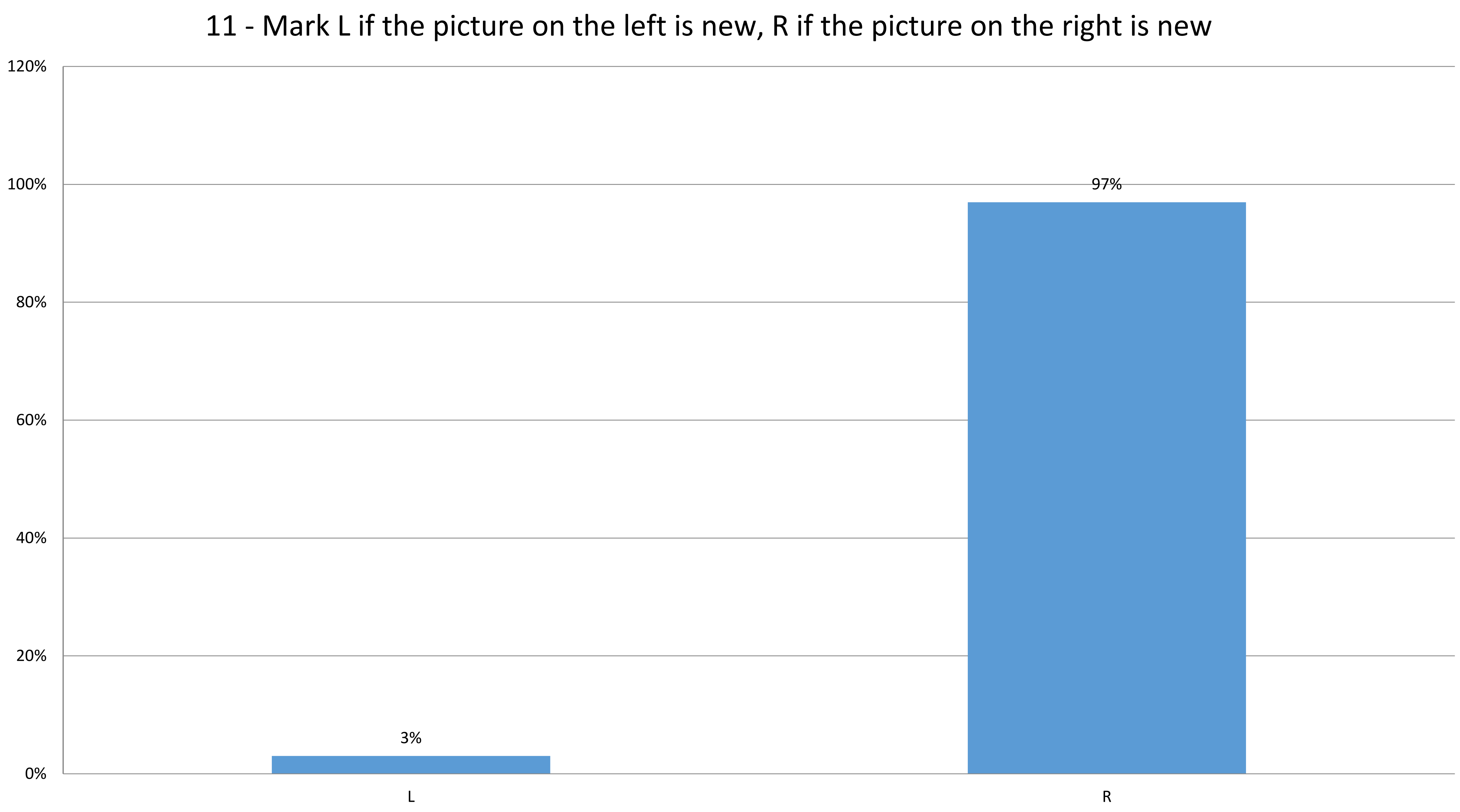
Mark L if the picture on the left is new, R if the picture on the right is new

10


10 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

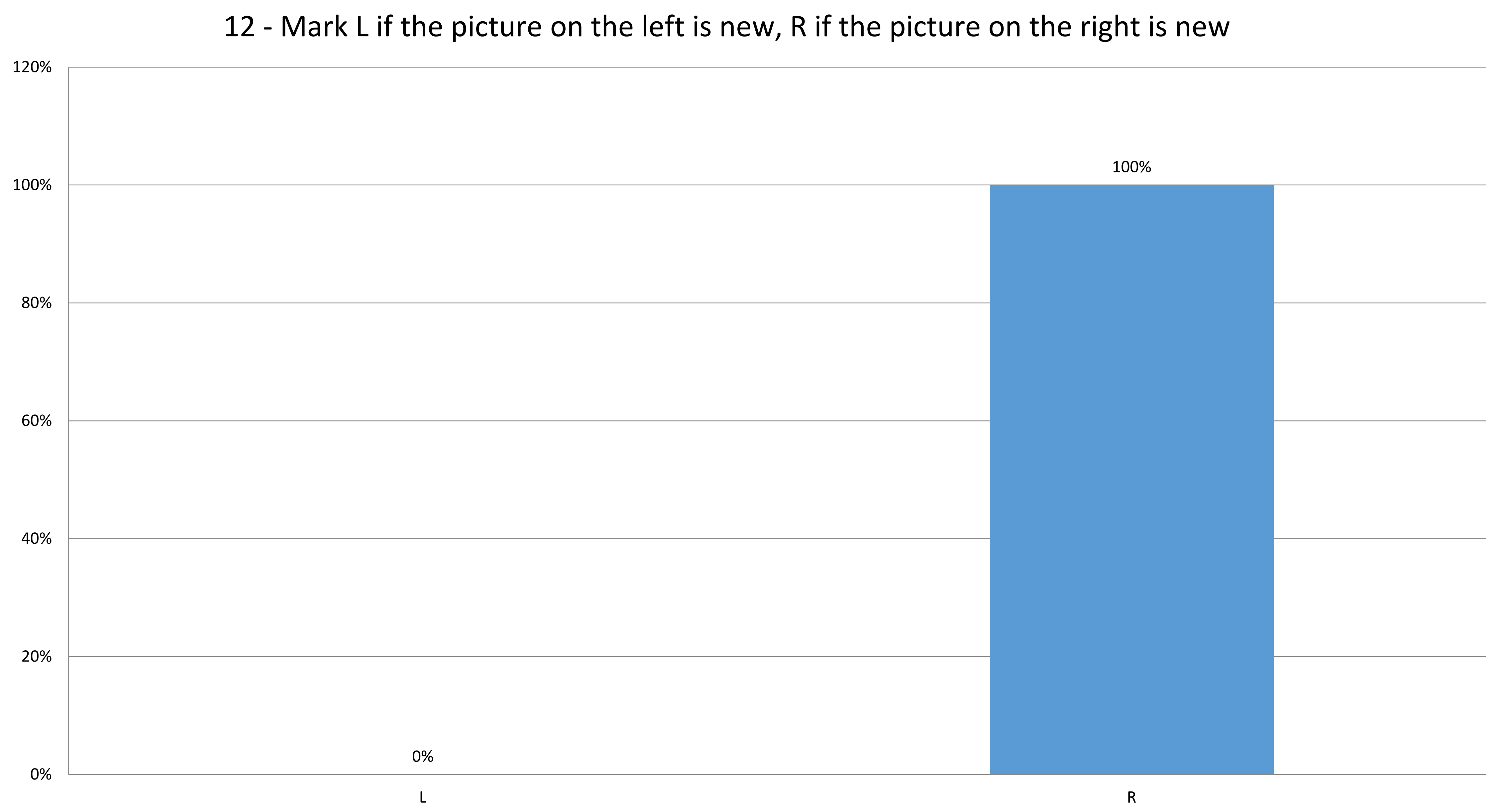
Mark L if the picture on the left is new, R if the picture on the right is new

11


11 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
3%
97%
L
R

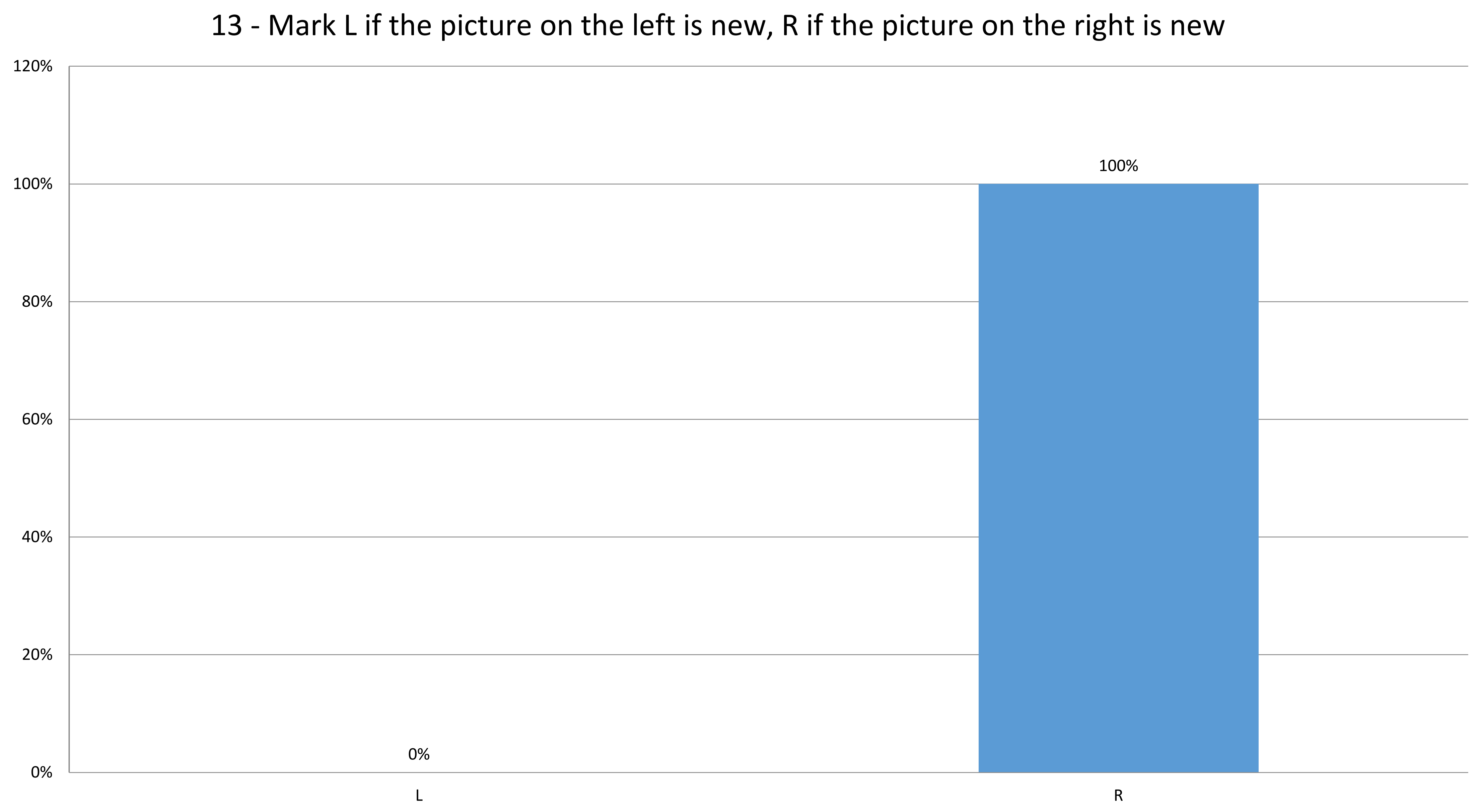
Mark L if the picture on the left is new, R if the picture on the right is new

12


12 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

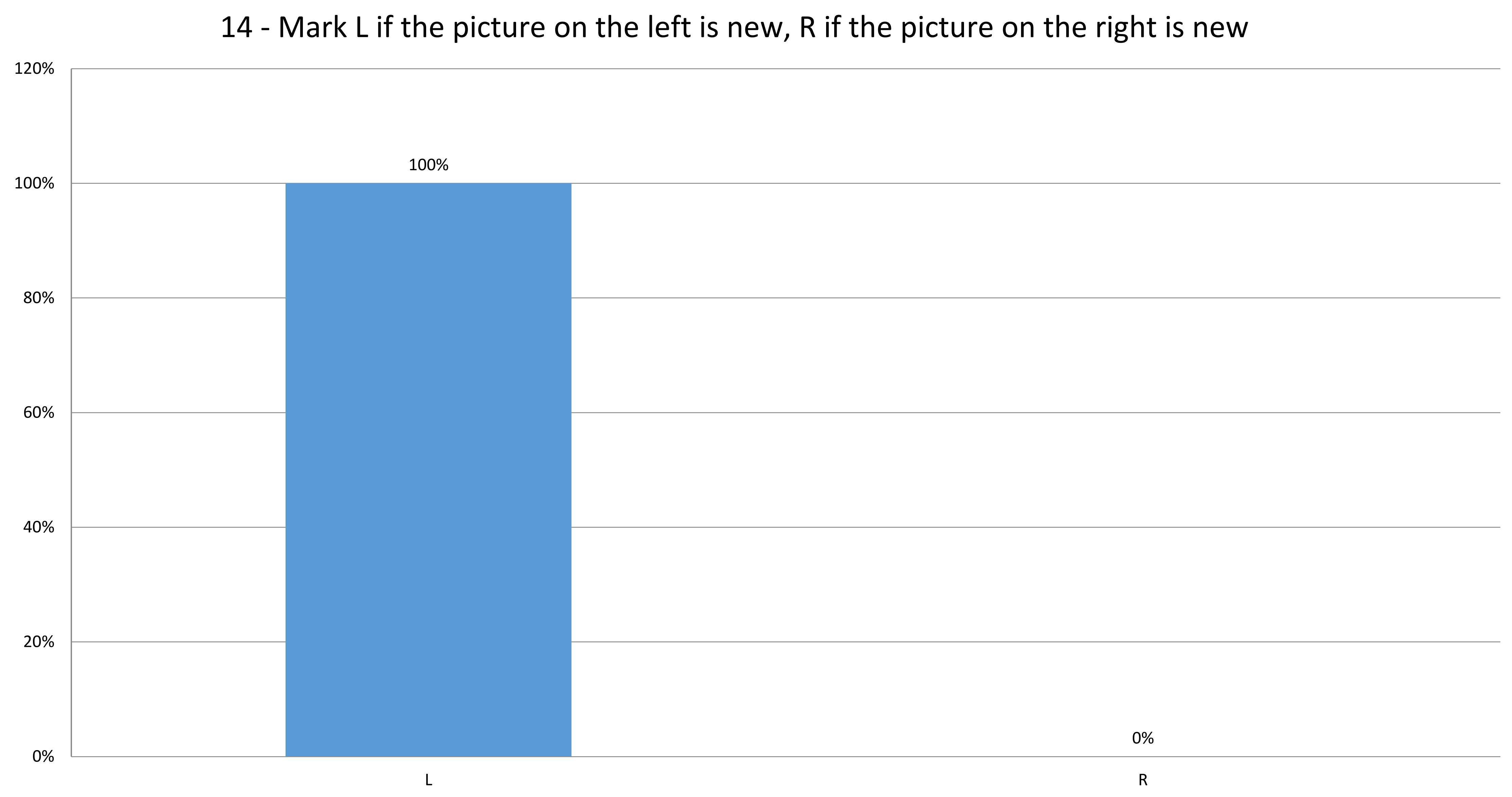
Mark L if the picture on the left is new, R if the picture on the right is new

13


13 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
0%
100%
L
R

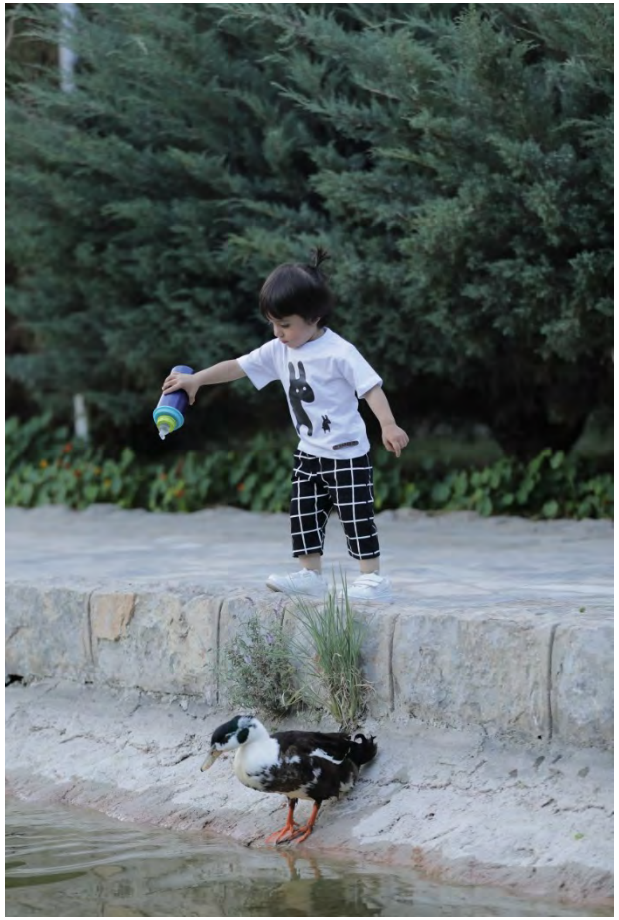
Mark L if the picture on the left is new, R if the picture on the right is new

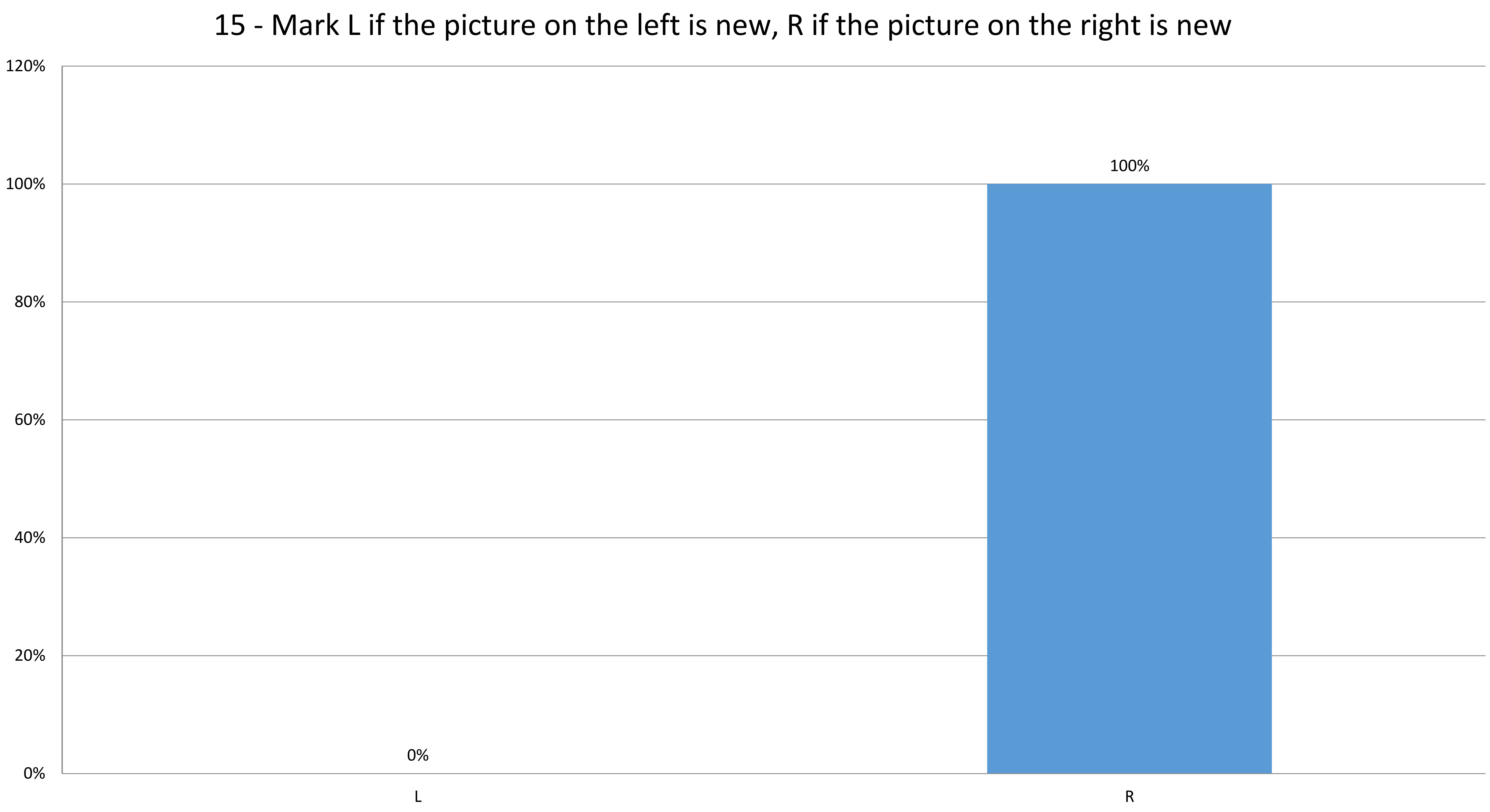
14


14 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

Mark L if the picture on the left is new, R if the picture on the right is new

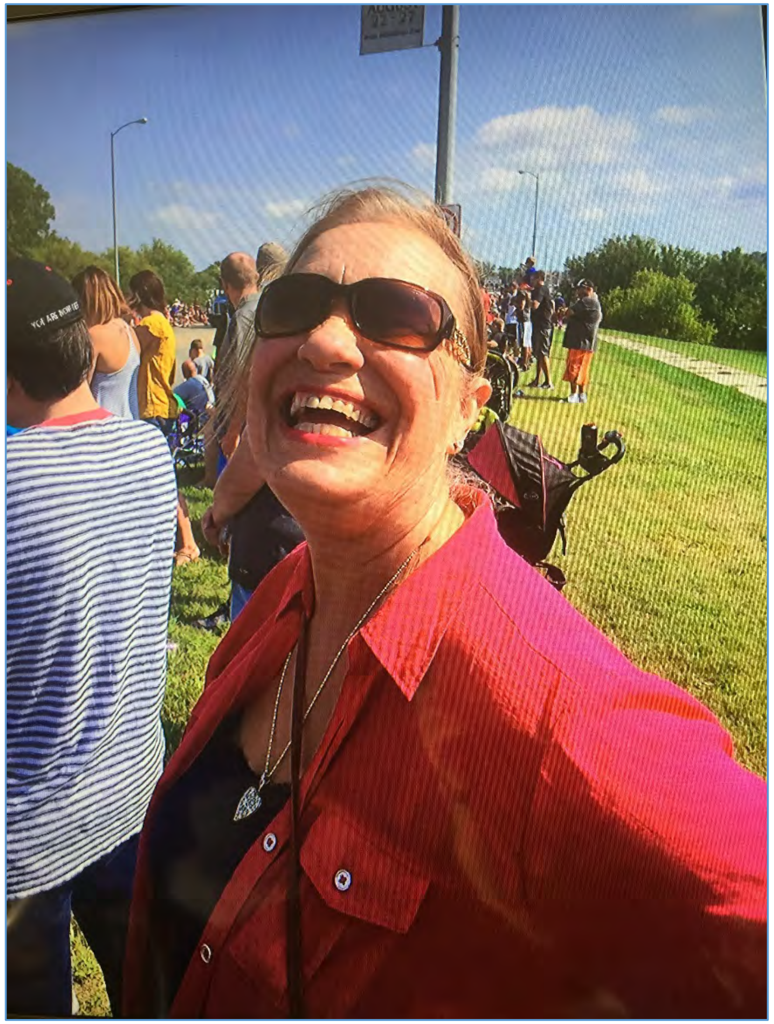
15

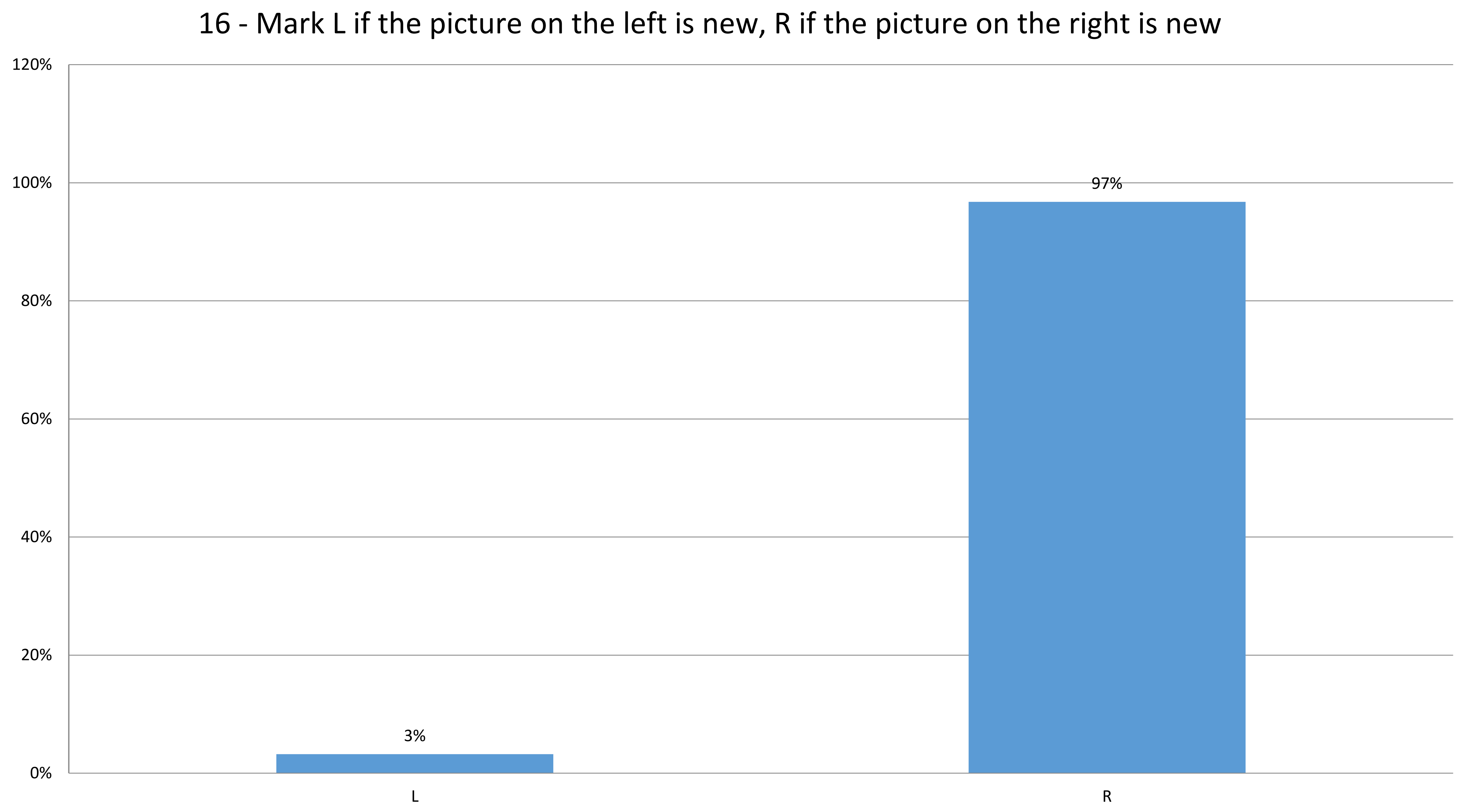



15 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

Mark L if the picture on the left is new, R if the picture on the right is new

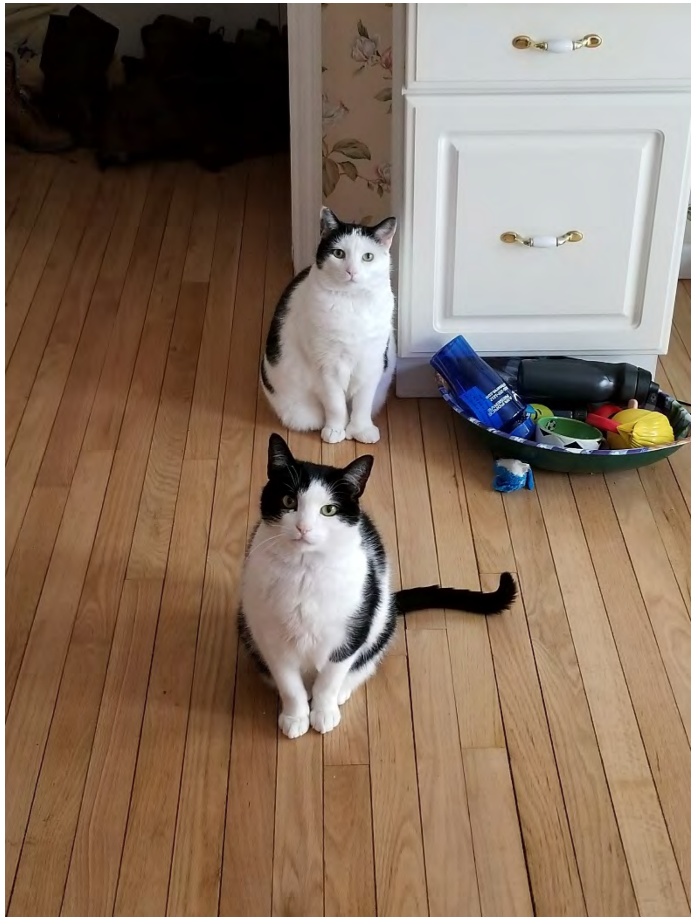
16

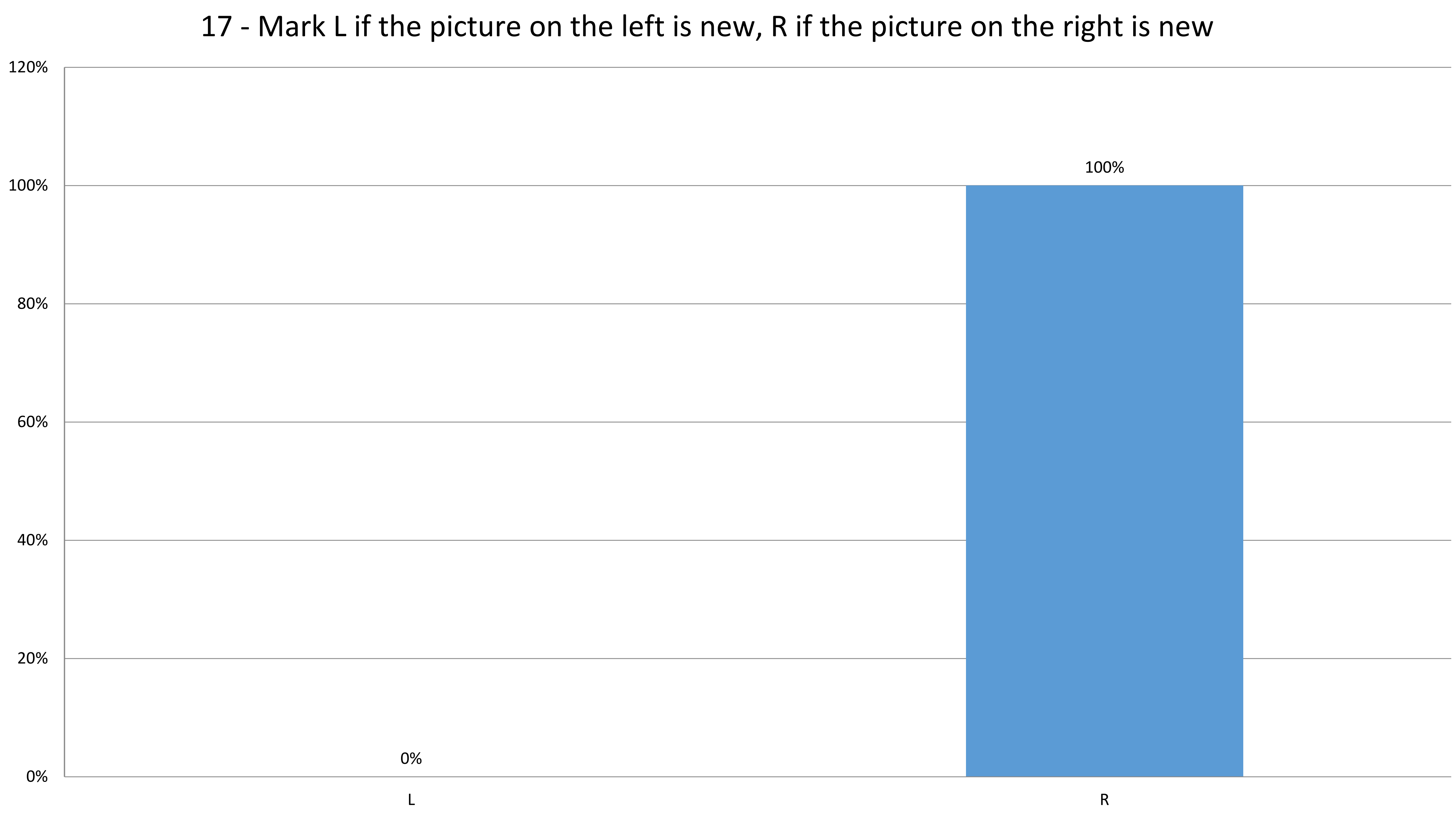



16 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
97%
3%
L
R

Mark L if the picture on the left is new, R if the picture on the right is new

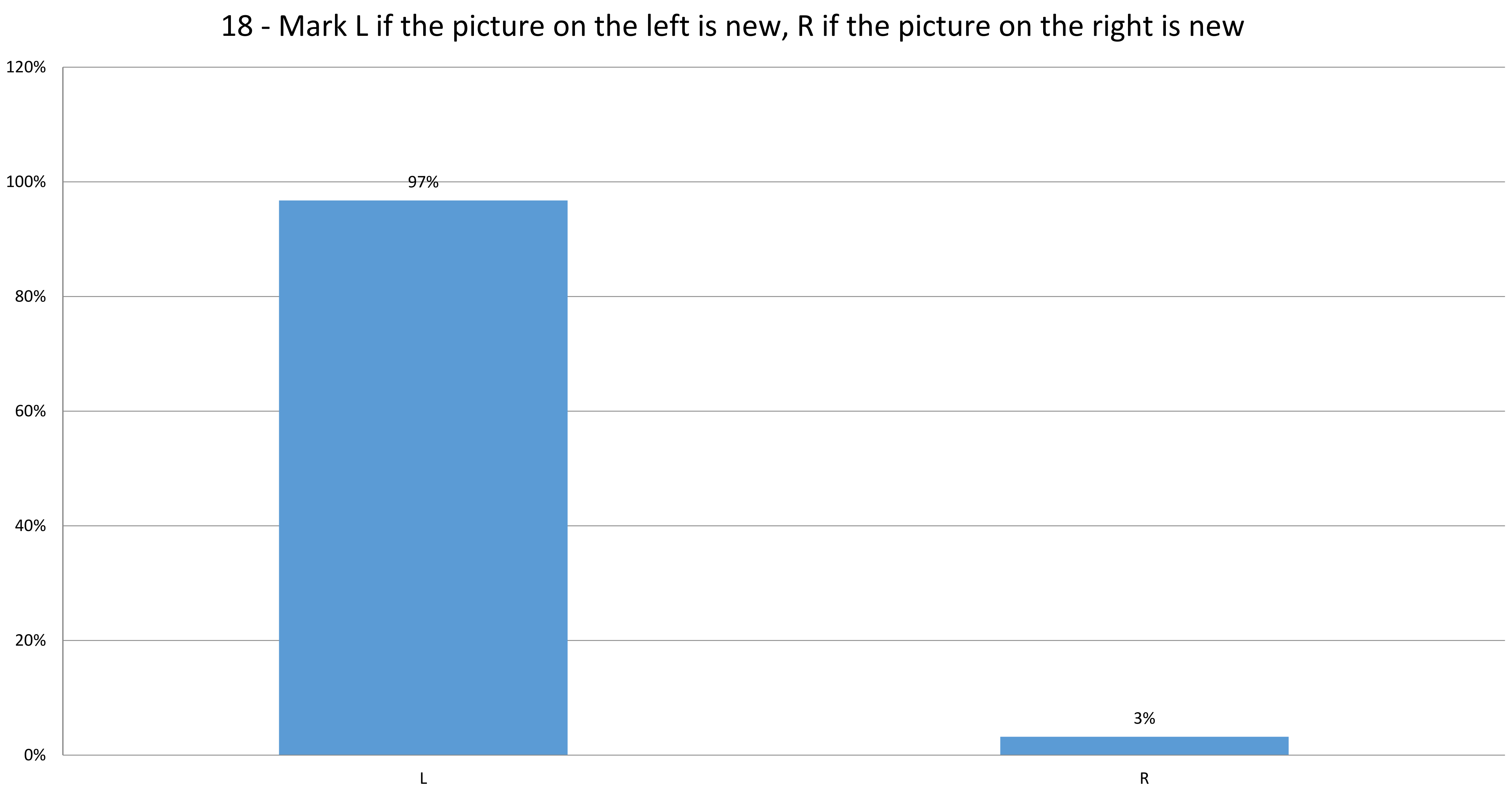
17




17 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

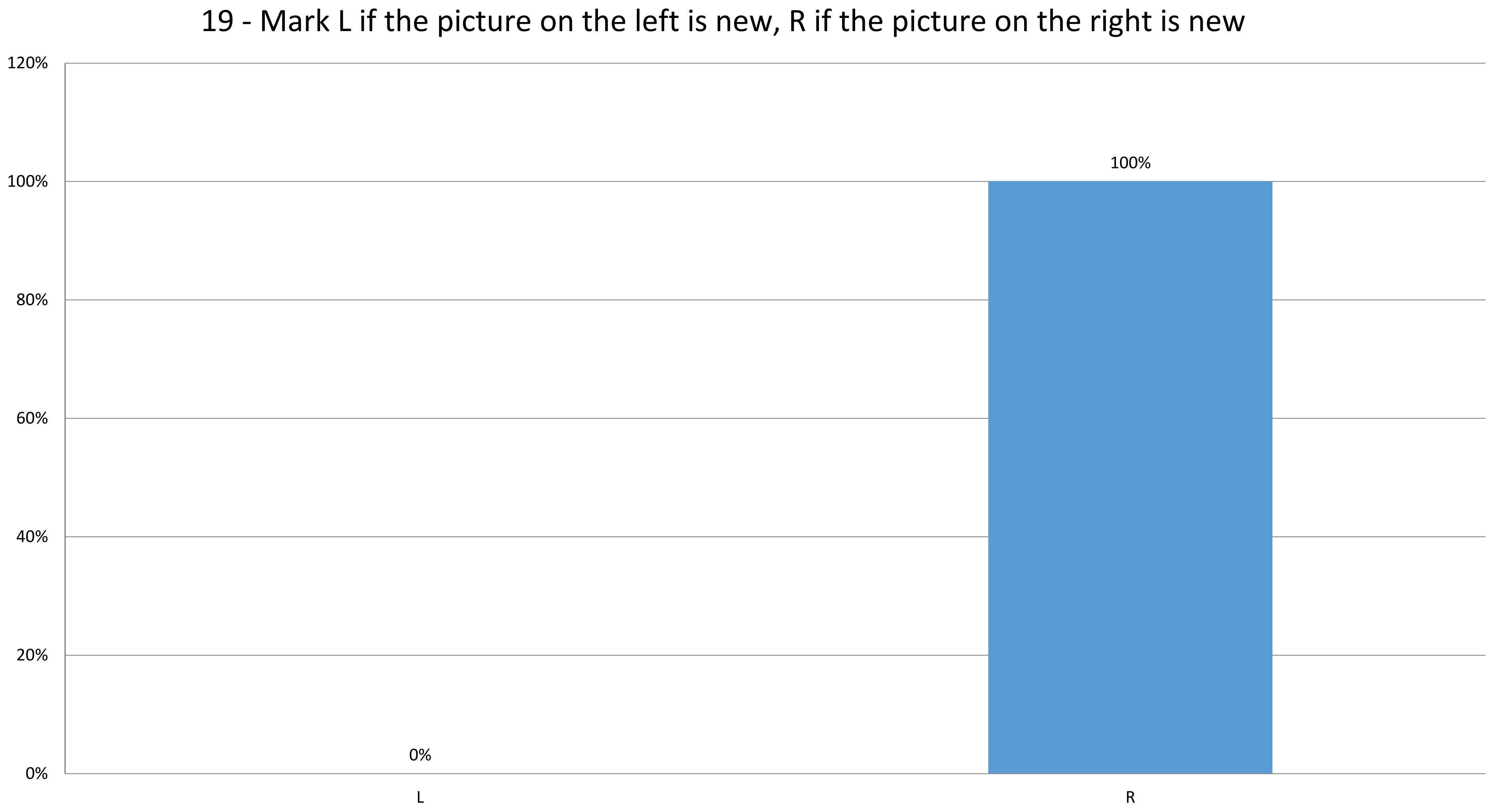
Mark L if the picture on the left is new, R if the picture on the right is new

18 


18 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
97%
3%
L
R

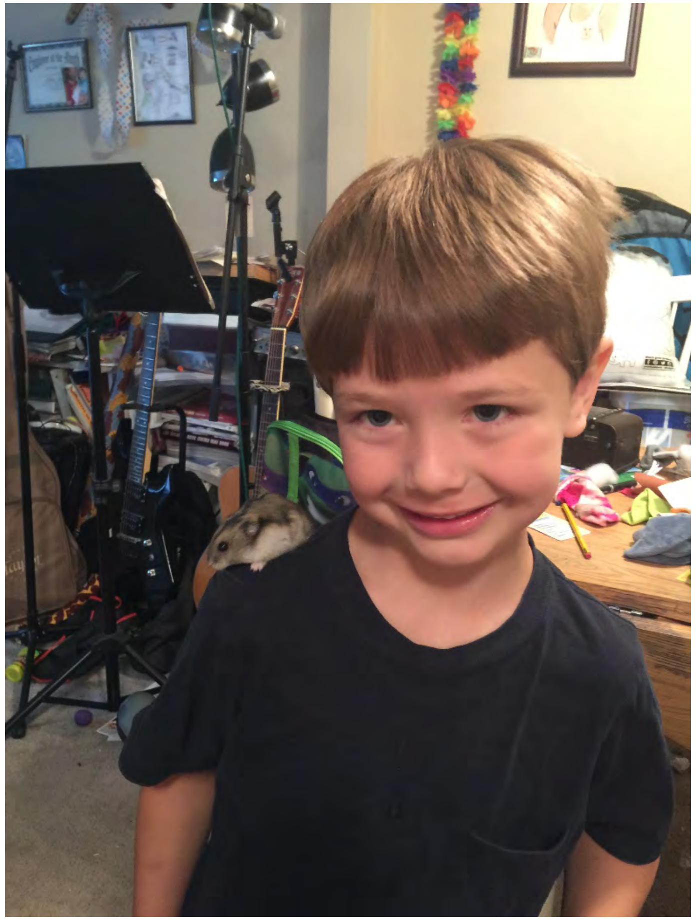
Mark L if the picture on the left is new, R if the picture on the right is new

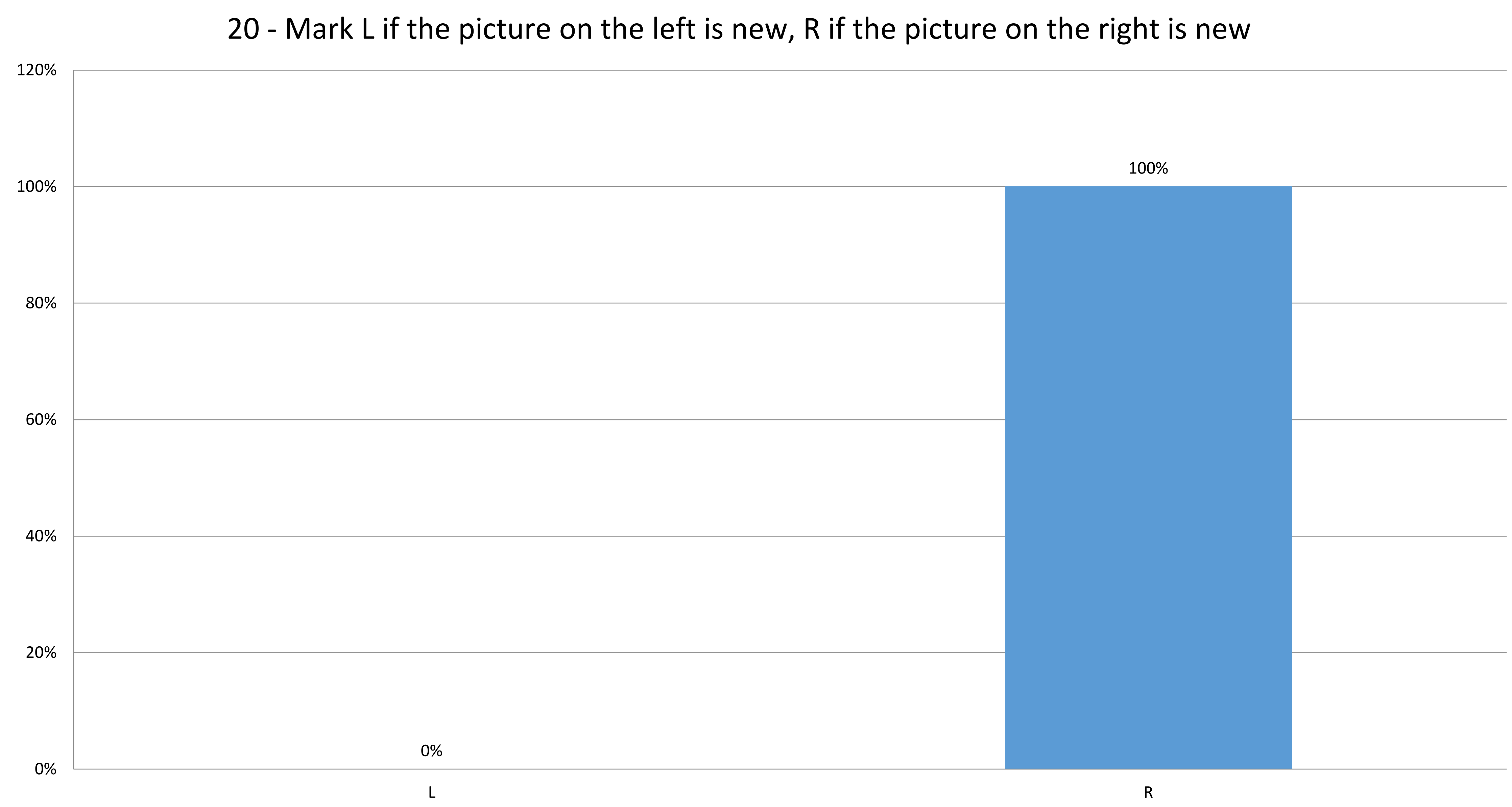
19


19 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

Mark L if the picture on the left is new, R if the picture on the right is new

20



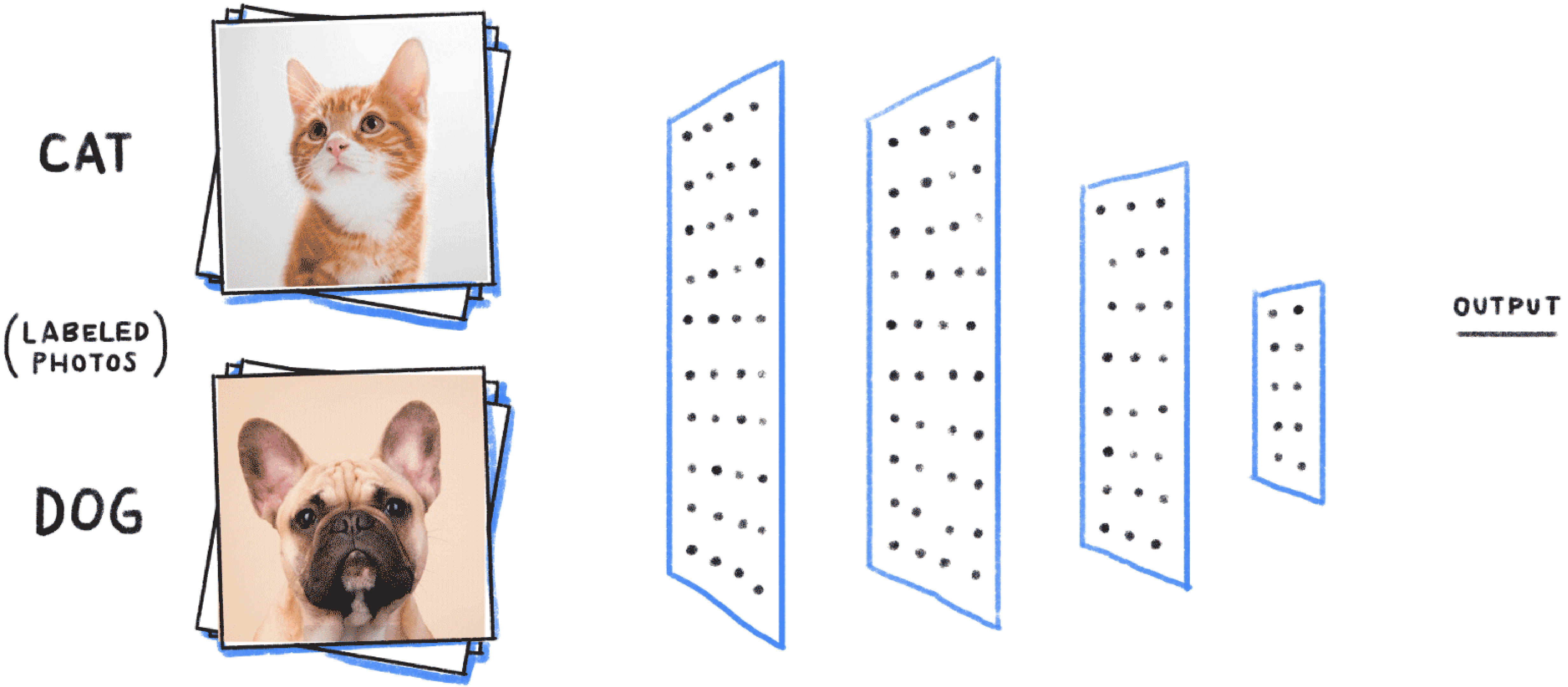

20 - Mark L if the picture on the left is new, R if the picture on the right is new
120%
100%
80%
60%
40%
20%
0%
100%
0%
L
R

How did we do?

According to professor Vishton, this test gets similar results with 100, 1,000, and even 3,000 slides.

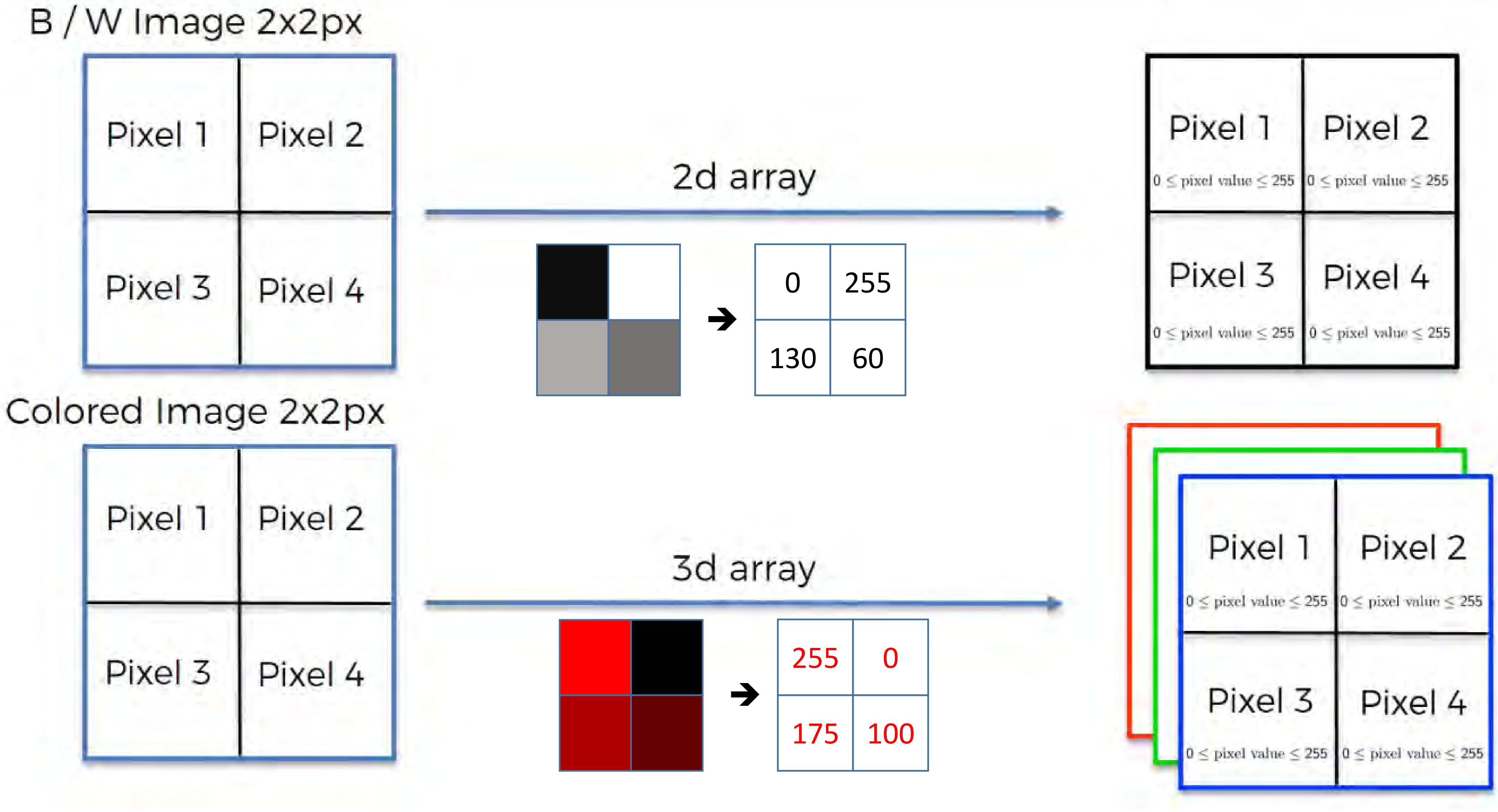
The capacity of the human mind to remember images and locations is seemingly unlimited.

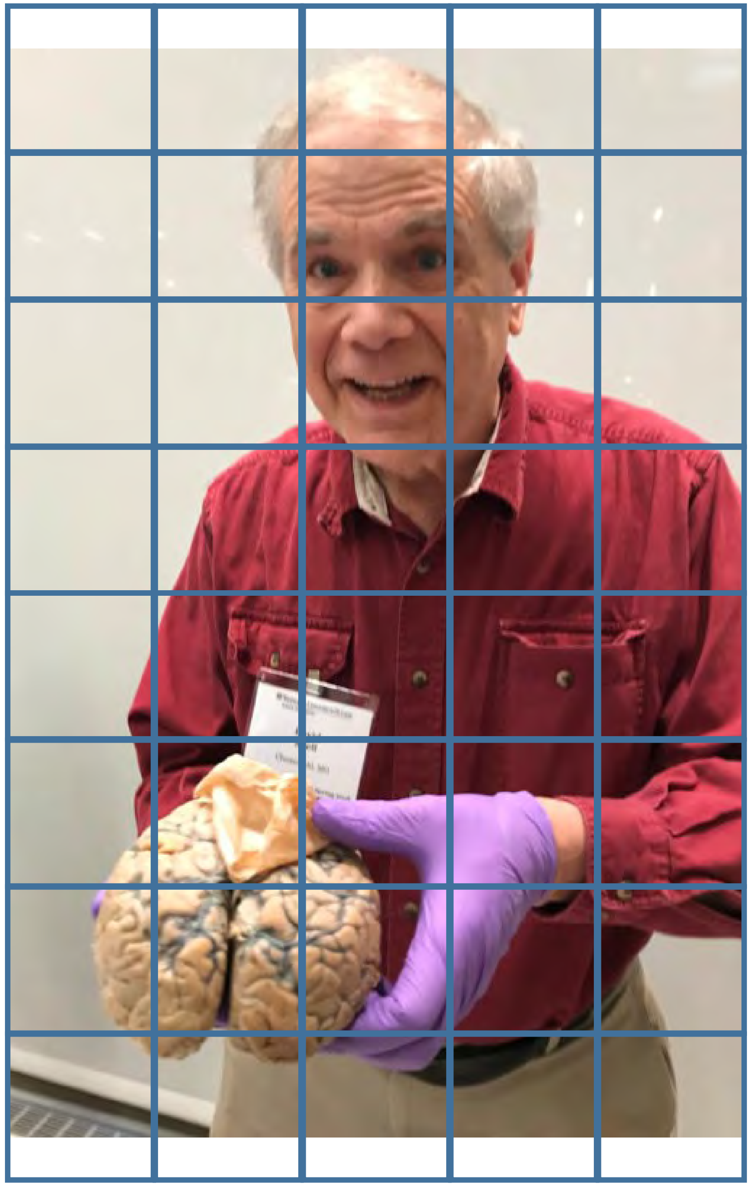
But how can an artificial neural network work with (identify) images?



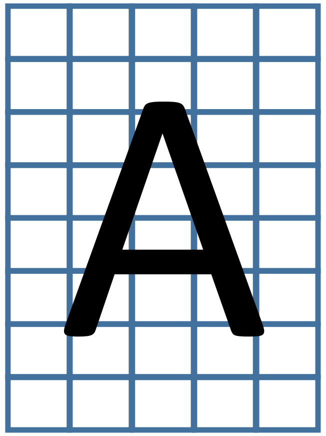
I found this gif on https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8 and several other websites but I am not sure of the actual source

Digitization – a monochrome picture can be represented as a collection of pixels of differing intensities of black/white/whatever; and a color picture is just a collection of similar monochrome layers – one for each primary color



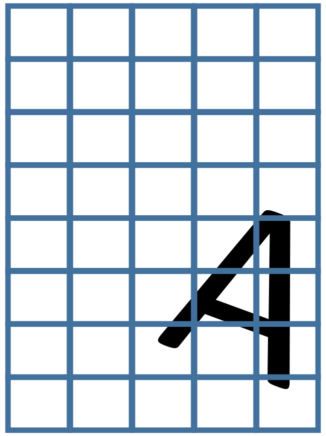


So, now that we know how to digitize a picture, why shouldn't we just place a grid over it and calculate our digital pattern?



Because for one thing, recognition is highly dependent upon everything being exactly aligned.

We need a way to account for changes in position, size, and perspective.



A convolution is a handy way to make our identification process more tolerant of real-world variances.

CONVOLUTION OPERATIONS

- In 1-D, convolution of two signals x[n] and h[n] is defined as the following formula:
  - (the asterisk denotes 'convolve', not times)
  - Can use a Flip, Shift, and Sum method

$$y[n] = x[n] * h[n]$$

$$\sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

The trick is through digitization and convolution operations. Here is the mathematical formula for a convolution. In this session, we will focus instead on a more intuitive approach.

Convolution – it reduces the dependence upon exact placement

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Pattern

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Pattern (Filter)

$=$

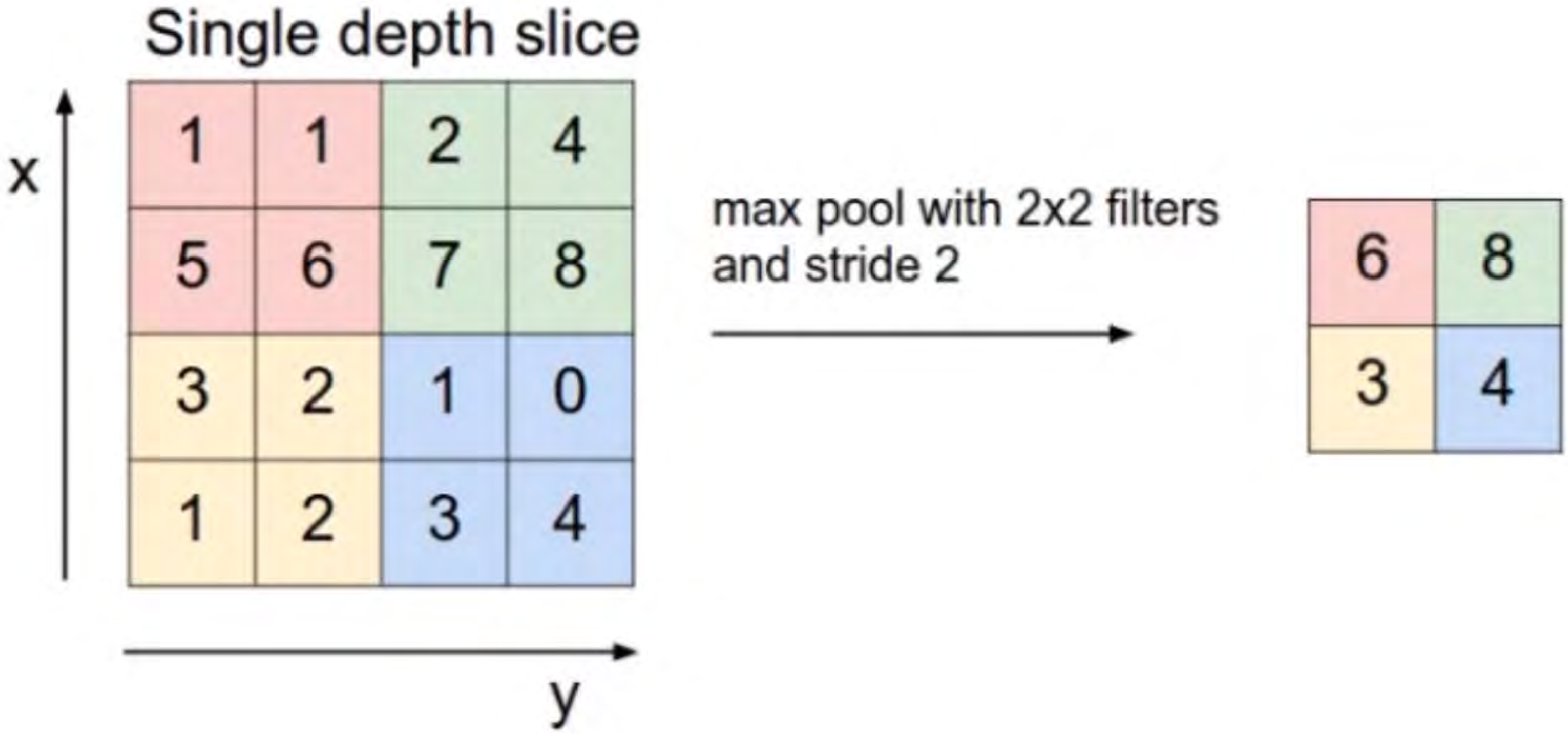
0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Output Pattern

$$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} \otimes \begin{bmatrix}0&0&1\\1&0&0\\0&1&1\end{bmatrix} = \boxed{0}$$

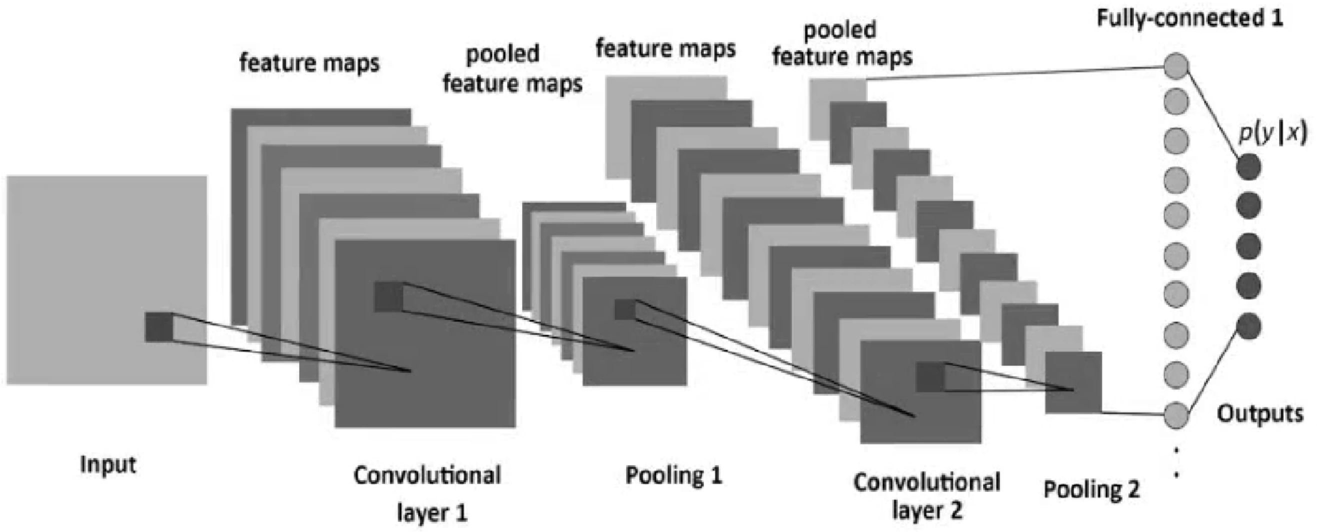
$$\begin{bmatrix}0&0&1\\1&0&0\\0&1&1\end{bmatrix} \otimes \begin{bmatrix}0&0&1\\1&0&0\\0&1&1\end{bmatrix} = \boxed{4}$$

Pooling

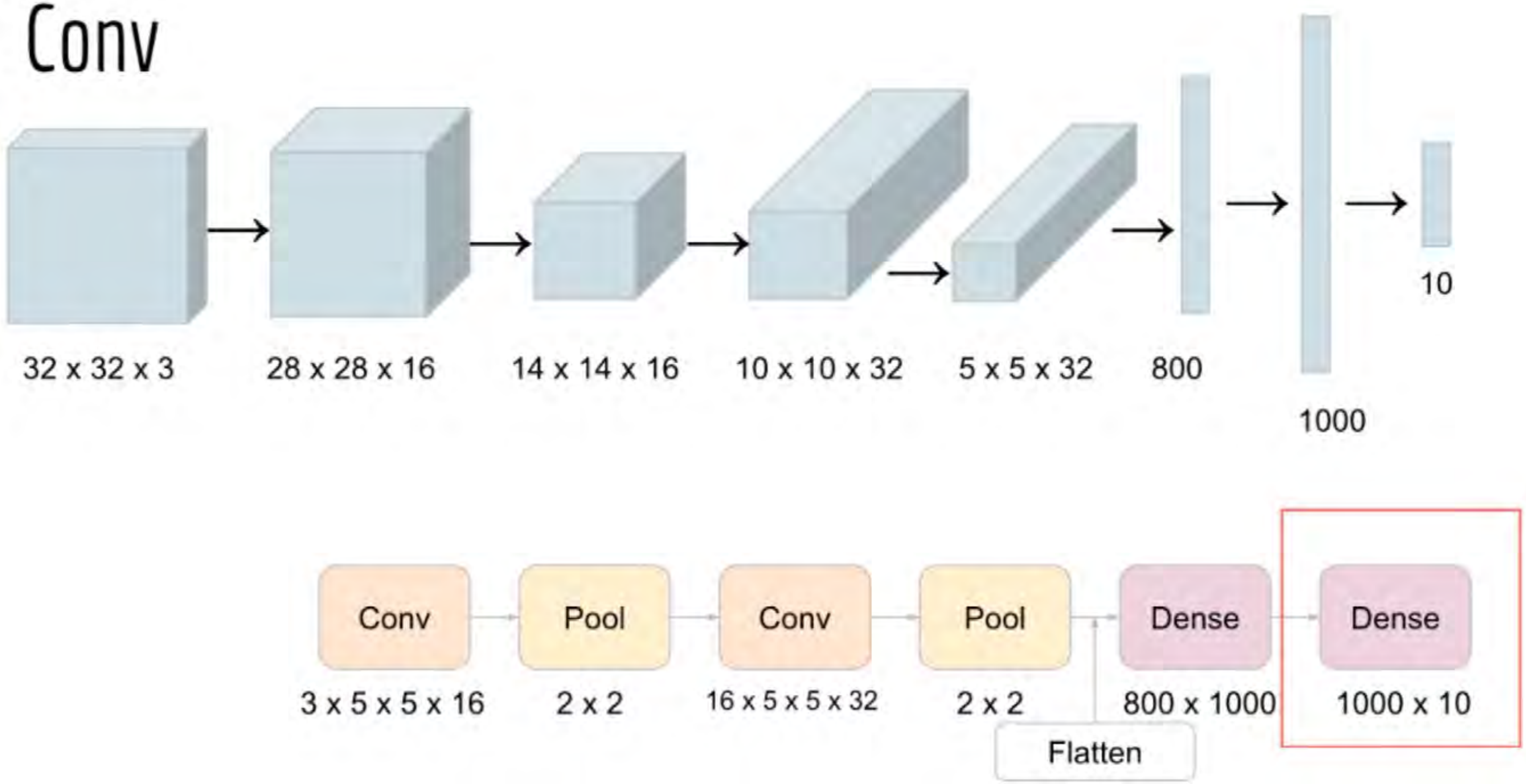

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

How do we choose filter sizes?

Simple way to get a feel for what others are doing: read literature.

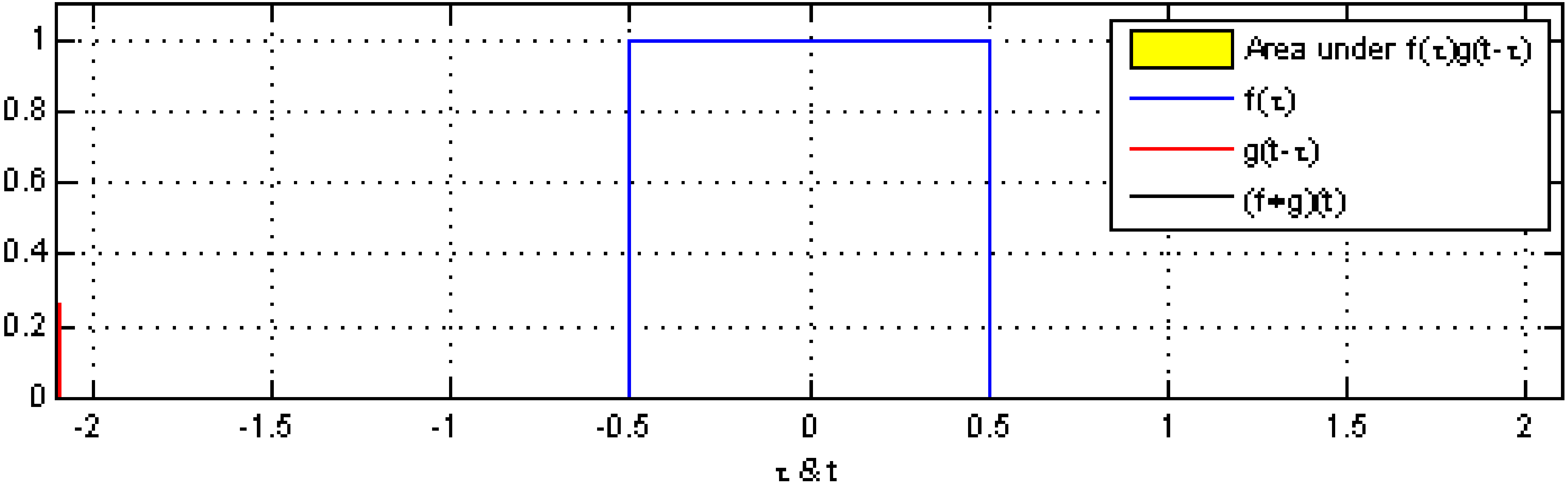


Conv

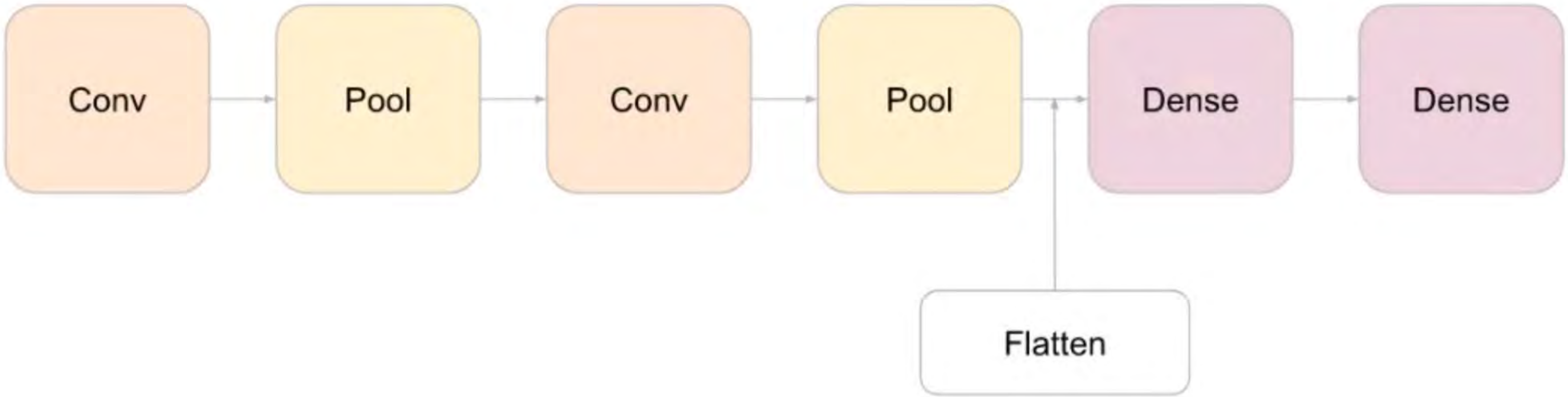


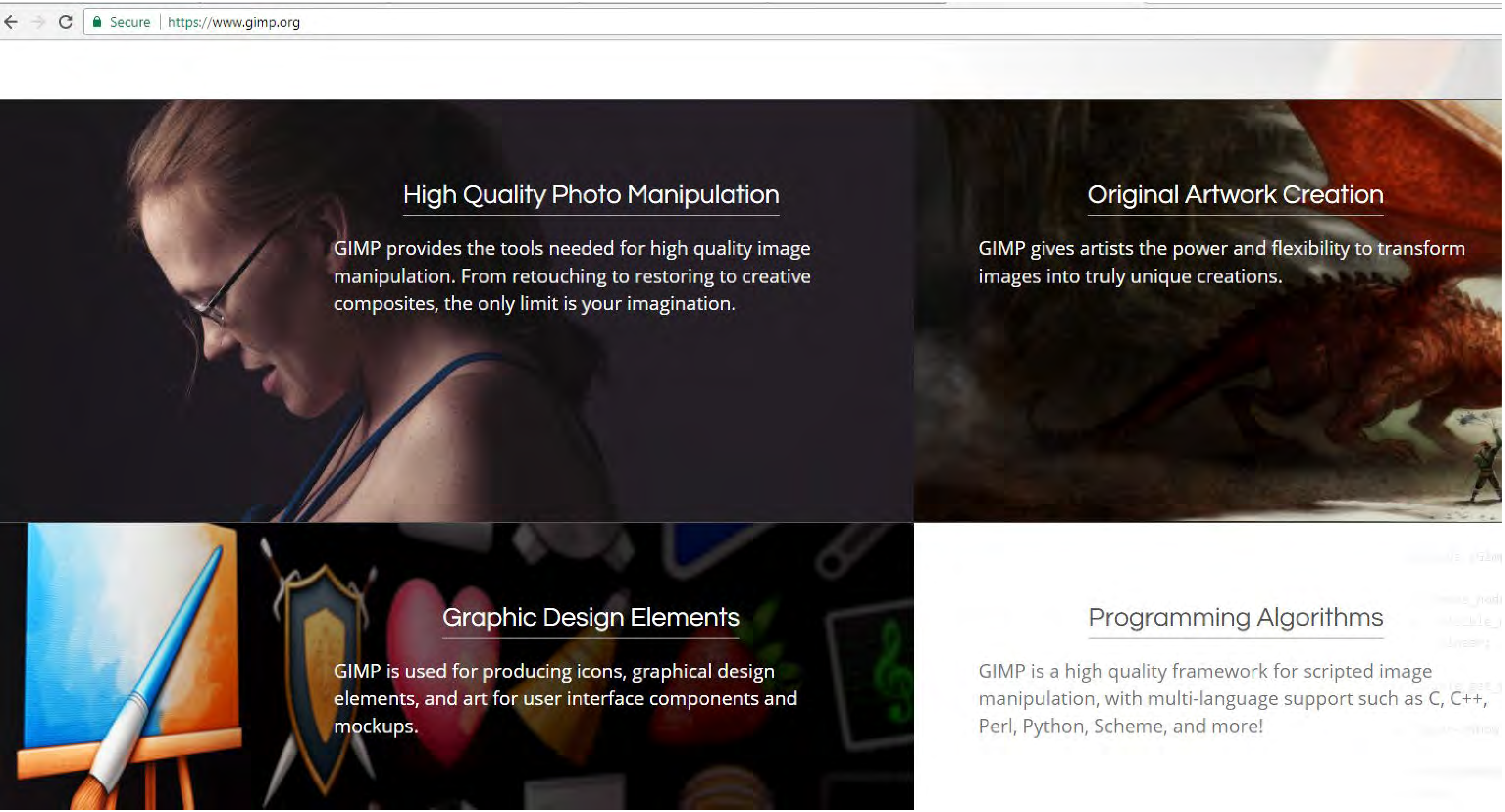
10
32 x 32 x 3
28 x 28 x 16
14 x 14 x 16
10 x 10 x 32
5 x 5 x 32
800
1000
Conv
Pool
Conv
Pool
Dense
Dense
3 x 5 x 5 x 16
2 x 2
16 x 5 x 5 x 32
2 x 2
800 x 1000
1000 x 10
Flatten

As an alternate perspective, here is a visualization
of the convolution of two box functions

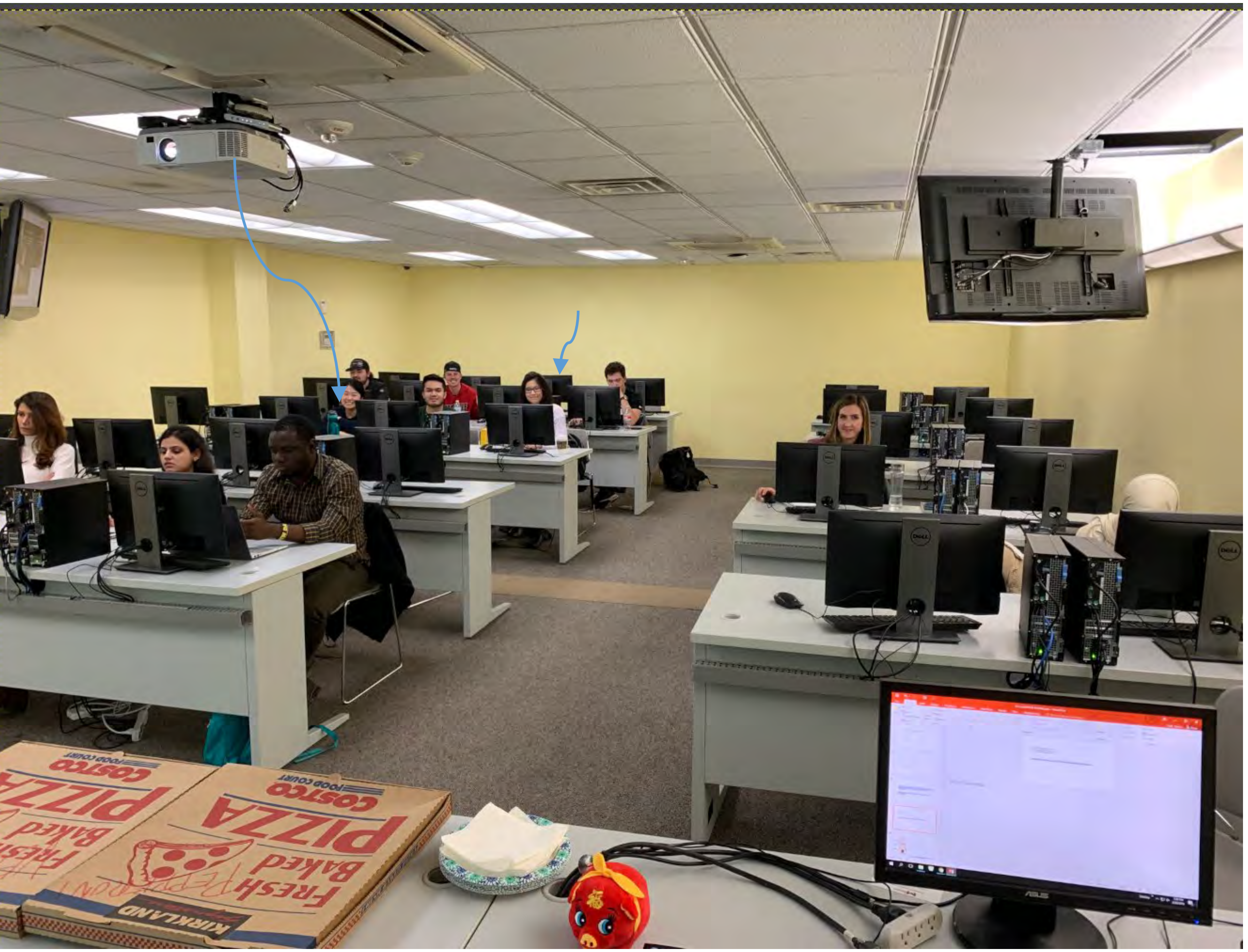


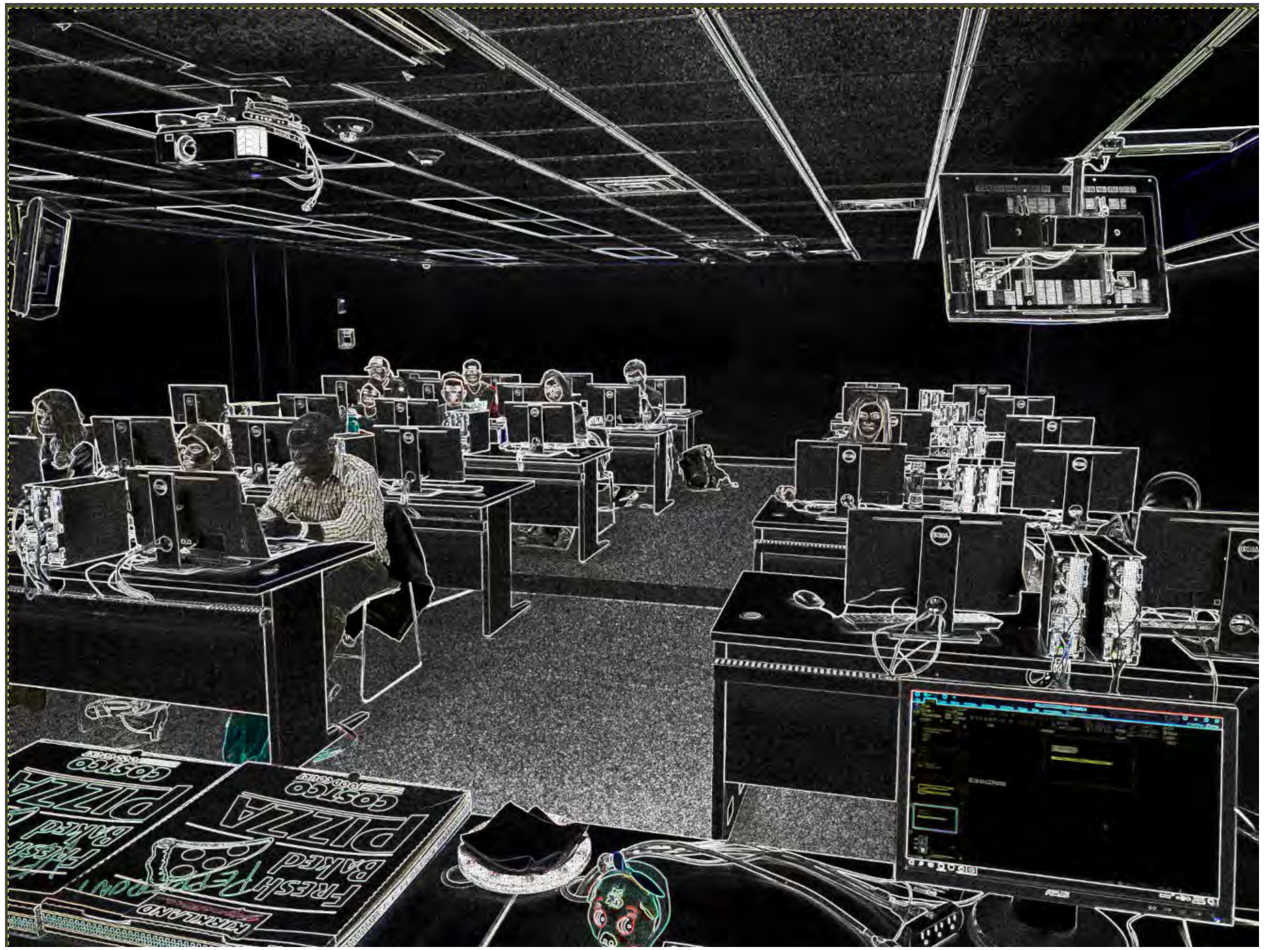
Full CNN





Here is a great FREE Convolutional Neural Network tool for you. https://www.gimp.org/

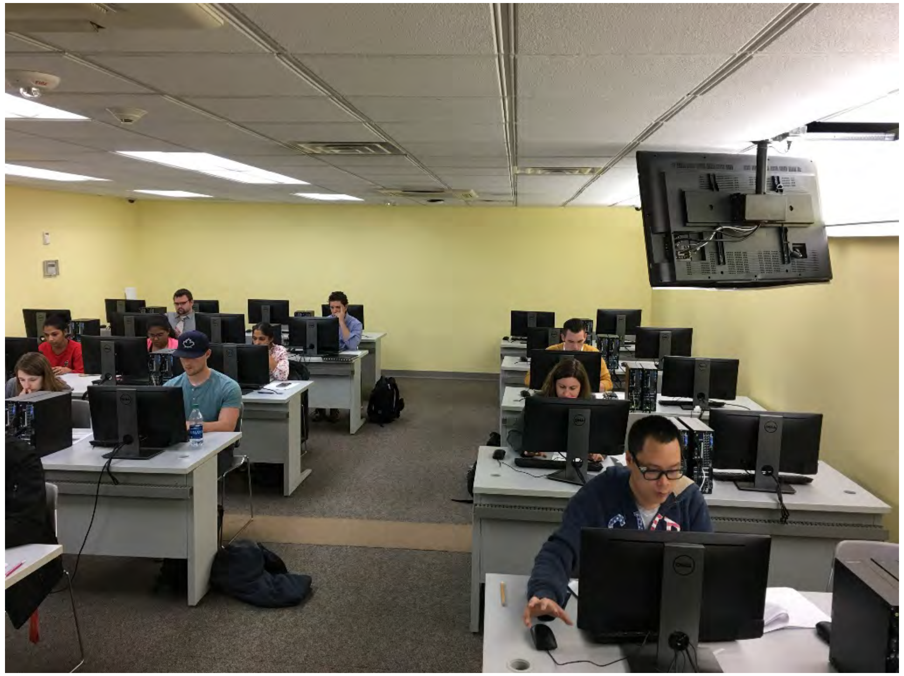


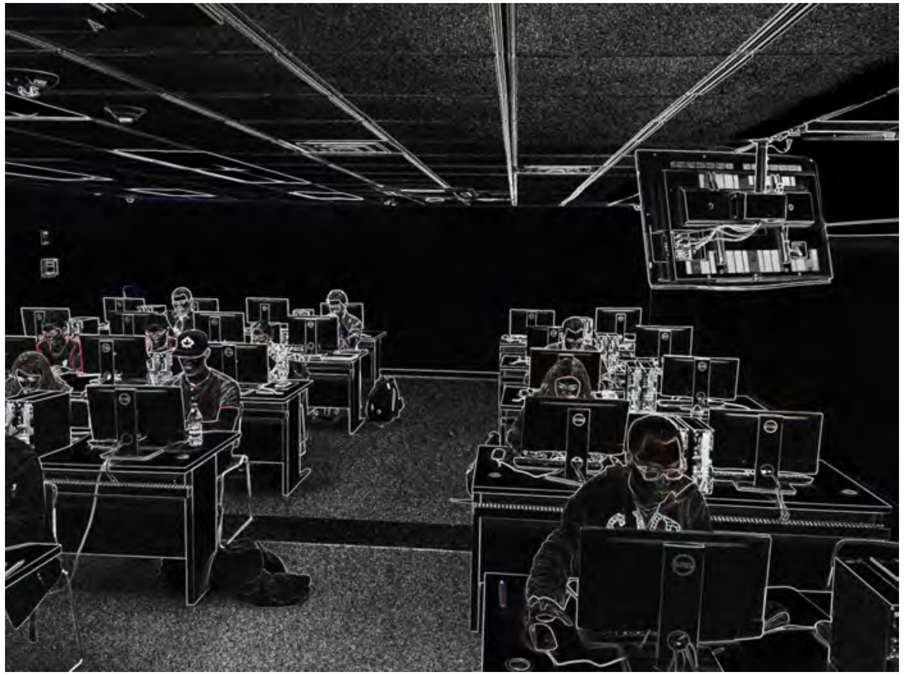


This edge detection took less than a second!

GIMP (and CNNs) are Magical!
www.gimp.org

ASSIGNMENT: Load the GIMP program and use it to detect the edges of a favorite picture. Hint: use Filters, Edge Detect, Edge.





Overview of Predictive Analytics Techniques

- Deep Learning

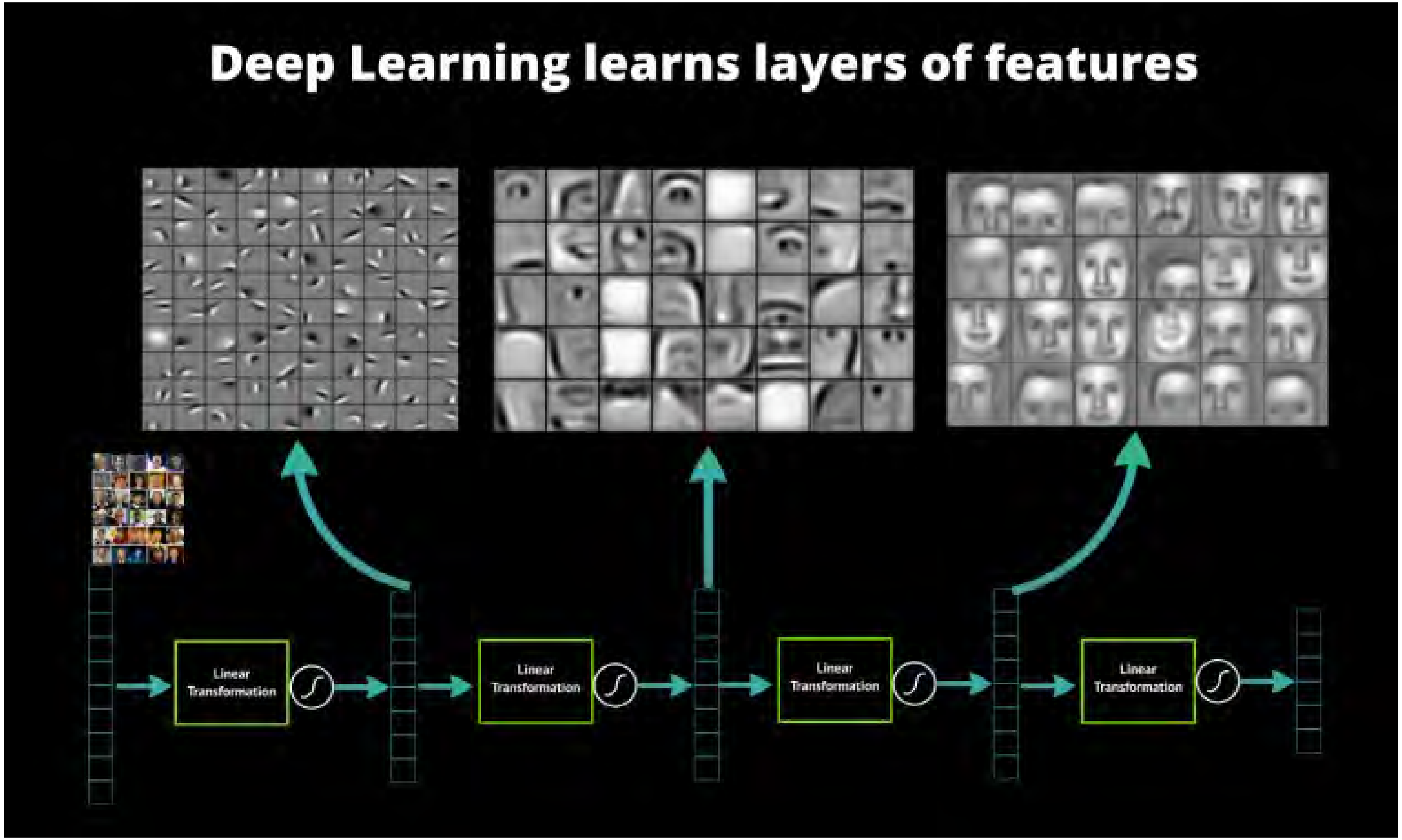
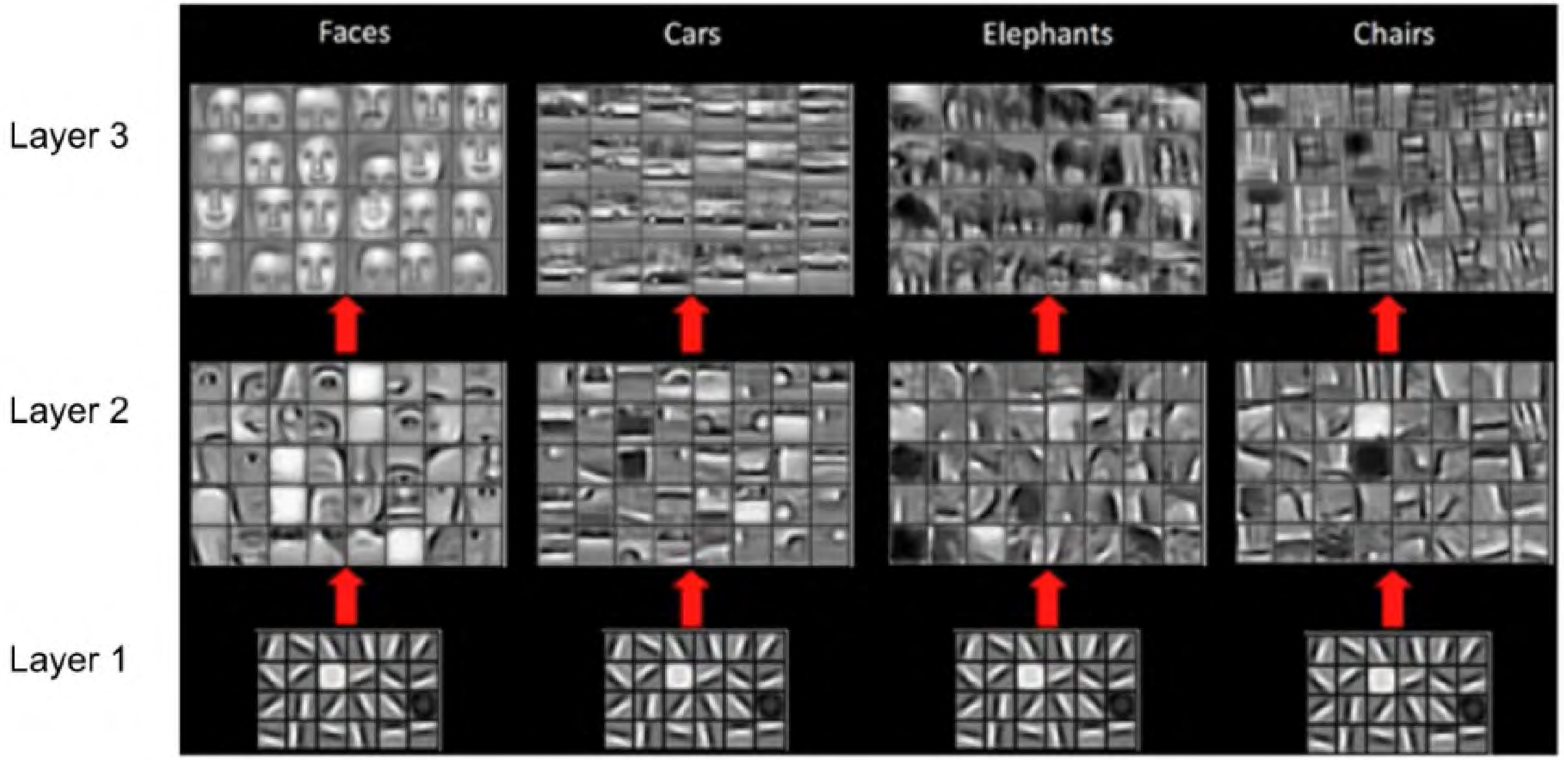


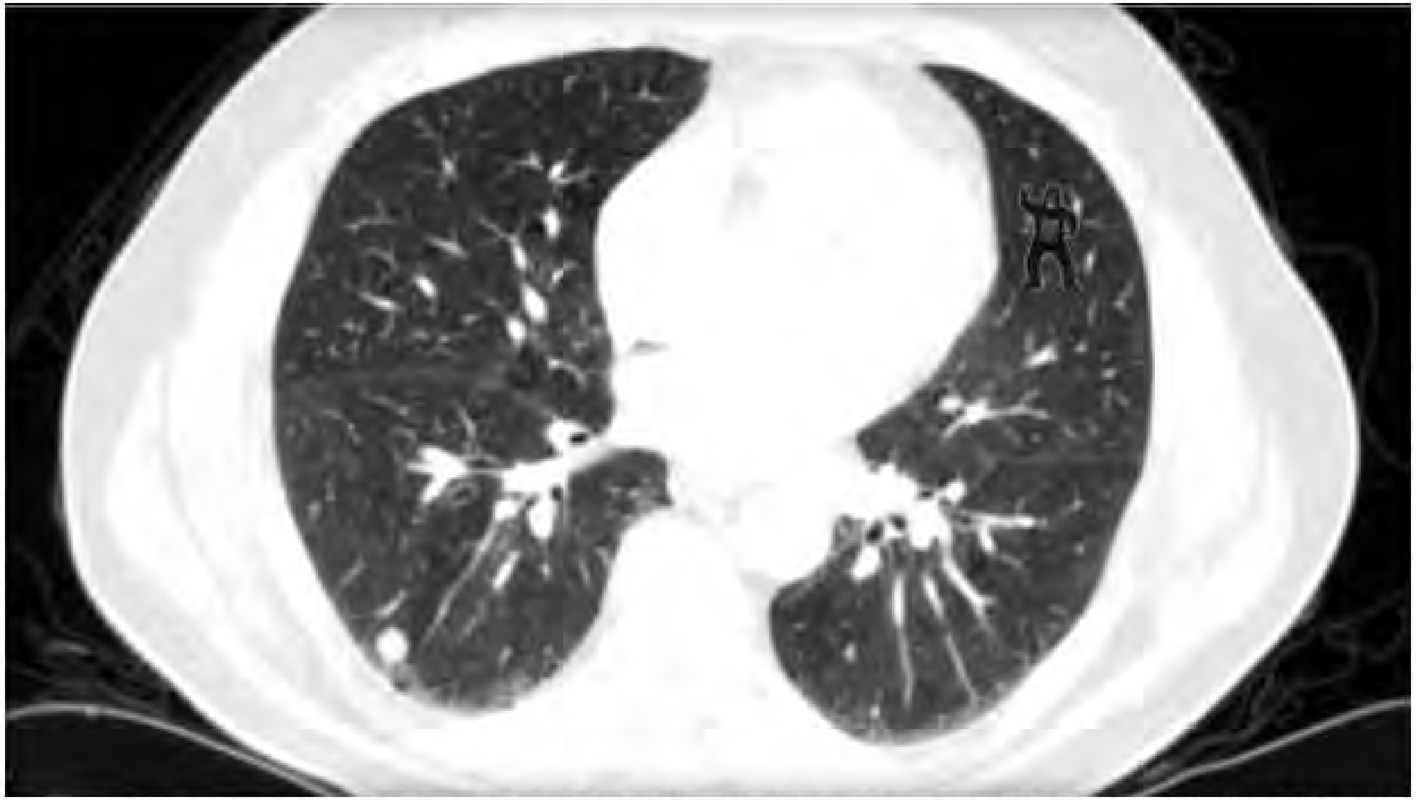
Image used with permission from DataRobot®: https://www.datarobot.com/blog/a-primer-on-deep-learning/

What has a CNN learned?



By MICHELLE CASTILLO | CBS NEWS | February 12, 2013, 3:02 PM

Can you spot the gorilla in this CT scan? Most radiologists couldn't



Do you see the gorilla in this scan? / MELISSA VO, AND JEREMY WOLFE/PSYCHOLOGICAL SCIENCE,TRAFTON DREW

Generative Adverserial Networks (GANs)

The Forgery Game: Generative Adversarial Networks

By Michael Niemerg

Imagine a not-too-distant future. You open your mailbox to find a pretty ordinary-seeming catalog. You start to flip through it. Inside, you find pictures of beautiful, smiling people. You see perfectly manicured lawns and perfect bedrooms. The catch: None of this is real. These images weren't even created using computer graphics. All these images were created by a model—by a generative adversarial network (GAN).[1,2] Don't believe this is possible? There are already images of fake people that look eerily realistic[3] and ways to manipulate an image to turn that smile into a frown.[4]

artwork but the curator also improves at spotting the real art apart from the forgeries.

GAN models are neural networks. While the relationship

https://www.soa.org/Library/Newsletters/Predictive-Analytics-and-Futurism/2018/april/2018-predictive-analytics-iss17-niemerg.pdf

CNNs and Generative Adversarial Networks (GANs) – not just for pictures

https://www.soa.org/Library/Newsletters/Predictive-Analytics-and-Futurism/2018/december/2018-predictive-analytics-iss19-duncan.pdf

Semi-Supervised Learning With Generative Adversarial Networks

By Jeff Heaton

Figure 1
Real or Fake?

In the world of machine learning, supervised and unsupervised are the two premier methodologies usually discussed. Most problems, and the models used to deal with them, are either classified as supervised or unsupervised. However, there are other types of models beyond supervised and unsupervised. Models that support semi-supervised and reinforcement learning are two model types that have lately been gaining considerable traction. In this two-part article series, we will look at semi-supervised learning. This article will begin by introducing semi-supervised learning and the generative adversarial network (GAN). The GAN, which is usually shown in conjunction with image rendering, will be demonstrated to have insurance industry applications. The second article will provide a more technical implementation of a semi-supervised GAN, using Keras and TensorFlow, for health care data.

AN INTRODUCTION TO THE GENERATIVE ADVERSARIAL NETWORKS

GANs have received a great deal of publicity lately for their ability to

shaped ears. The background is usually a giveaway as well. The background of a GAN-generated image is typically surreal looking. It looks natural, but you are never quite sure what you are looking at. Linear projections that begin on one side of the face often do not align to what is behind the other side. For this image, the background is not that surreal, but I am also not entirely sure what I would classify the background as either. ... mon giveaways as well, particularly if the

Jeff Heaton, Ph.D., is vice president and data scientist at RGA Reinsurance Company Inc. He can be reached at *jheaton@rgare.com*.

The Possible Role of Convolutional Neural Networks in Mortality Risk Prediction

By Holden Duncan

How might a computer tell the difference between the road and a tree while steering a car at 40 miles per hour? A rules system for each possible object that might be encountered could be created; however, such a system only allows for a limited amount of features, and is only as effective as the rules themselves. Increasing the number of rules might make for a more accurate classification, but such an engine would become unmanageable. In addition, attempting to hand-engineer such features limits a model to human intuition of pixel-by-pixel photo recognition.

There are already articles about random forests or linear/logistic regressions, but these methods involve the use of functions and hand-engineered features composed of different categories. As such, these models are generally unable to identify wholly new ideas without specific encoding. However, a model that excels in using the spatial relationship between complex features in data to generate accurate classification could learn to recognize new patterns. A convolutional neural network, or CNN, learns to use concrete low-level features in order to extract identifying

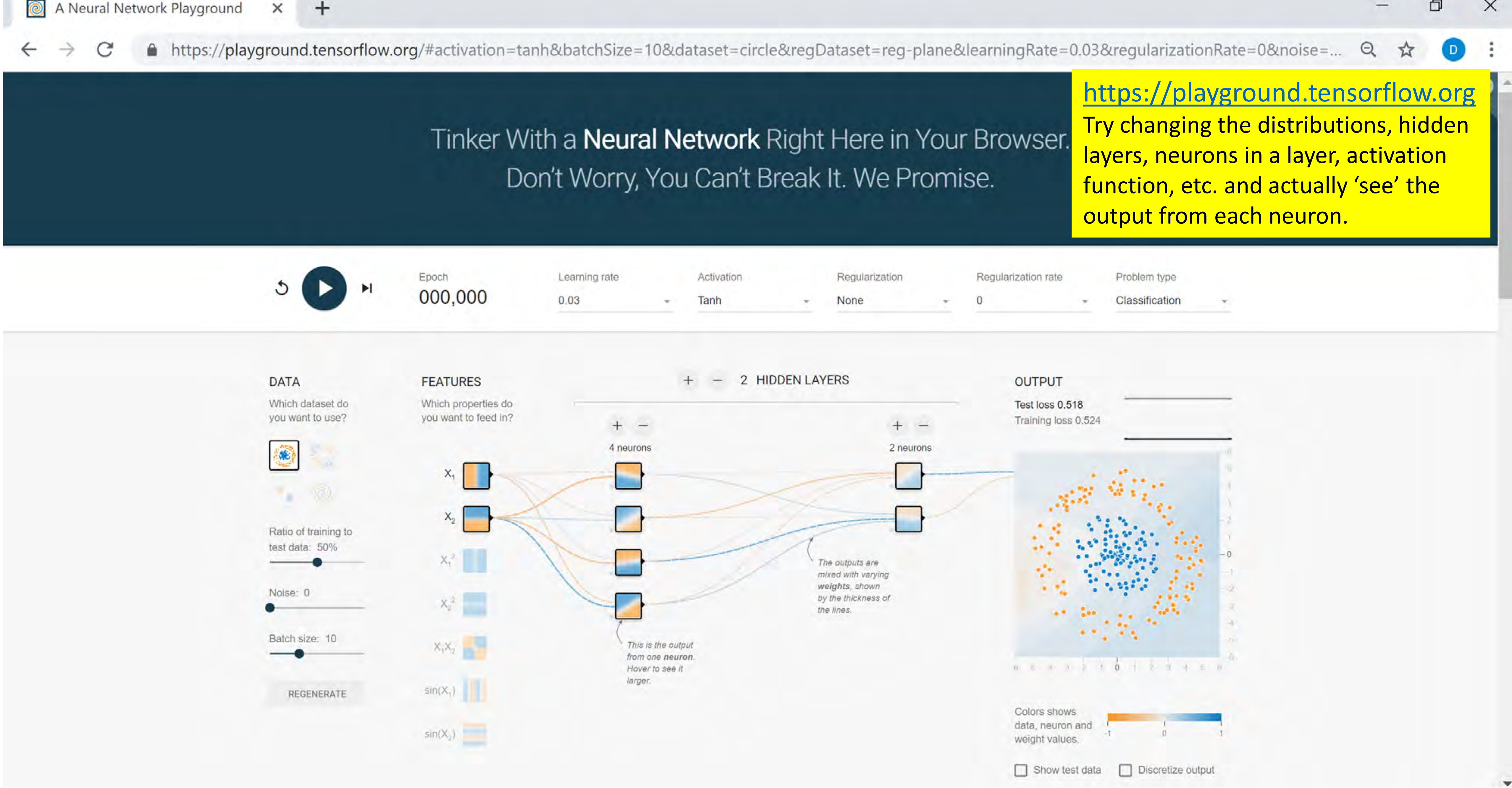
- Each node in layer A has a weighted connection to every node in layer B.
- Let A_x represent a given node in layer A, similarly for B_y in B, and let $Weight_{xy}$ be the weight of the connection between those two nodes. Then the input from A_x to B_y may be expressed as:

$$Ax_{out} * Weight_{xy}$$

Holden Duncan is a data scientist at RGA Reinsurance Company, in Chesterfield, Mo. He can be reached at *HoldenDDuncan@gmail.com*

https://playground.tensorflow.org

Try changing the distributions, hidden layers, neurons in a layer, activation function, etc. and actually ‘see’ the output from each neuron.



2019 PREDICTIVE ANALYTICS SYMPOSIUM

Generative Adversarial Networks

Balathasan Giritharan

Data Scientist

Bayer Crop Science

giritharanb@gmail.com

SOCIETY OF ACTUARIES®

CBS NEWS October 26, 2018, 8:33 AM

This AI-generated portrait just sold for a stunning $432,500

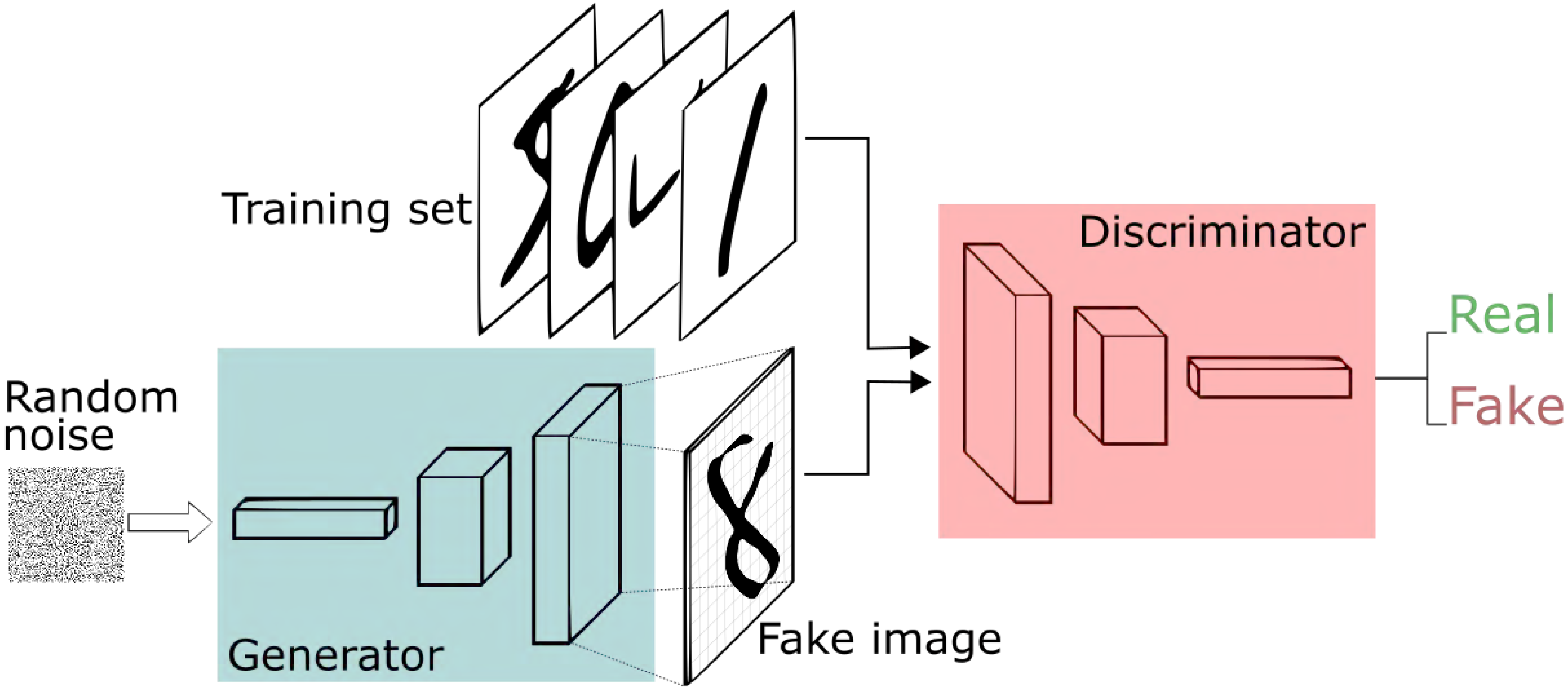
Is artificial intelligence set to become art's next medium?

AI artwork sells for $432,500 — nearly 45 times its high estimate — as Christie's becomes the first auction house to offer a work of art created by an algorithm

References & Sources

- Generative Adversarial Nets, Ian J. Goodfellow et al
  - https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
- An intuitive introduction to Generative Adversarial Networks (GANs)
  - https://medium.freecodecamp.org/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394
- A Beginner's Guide to Generative Adversarial Networks
  - https://skymind.ai/wiki/generative-adversarial-network-gan
- GANs from scratch:
  - https://medium.com/ai-society/gans-from-scratch-1-a-deep-introduction-with-code-in-pytorch-and-tensorflow-cb03cdcdba0f
- GAN Lab:
  - https://poloclub.github.io/ganlab/

General Architecture



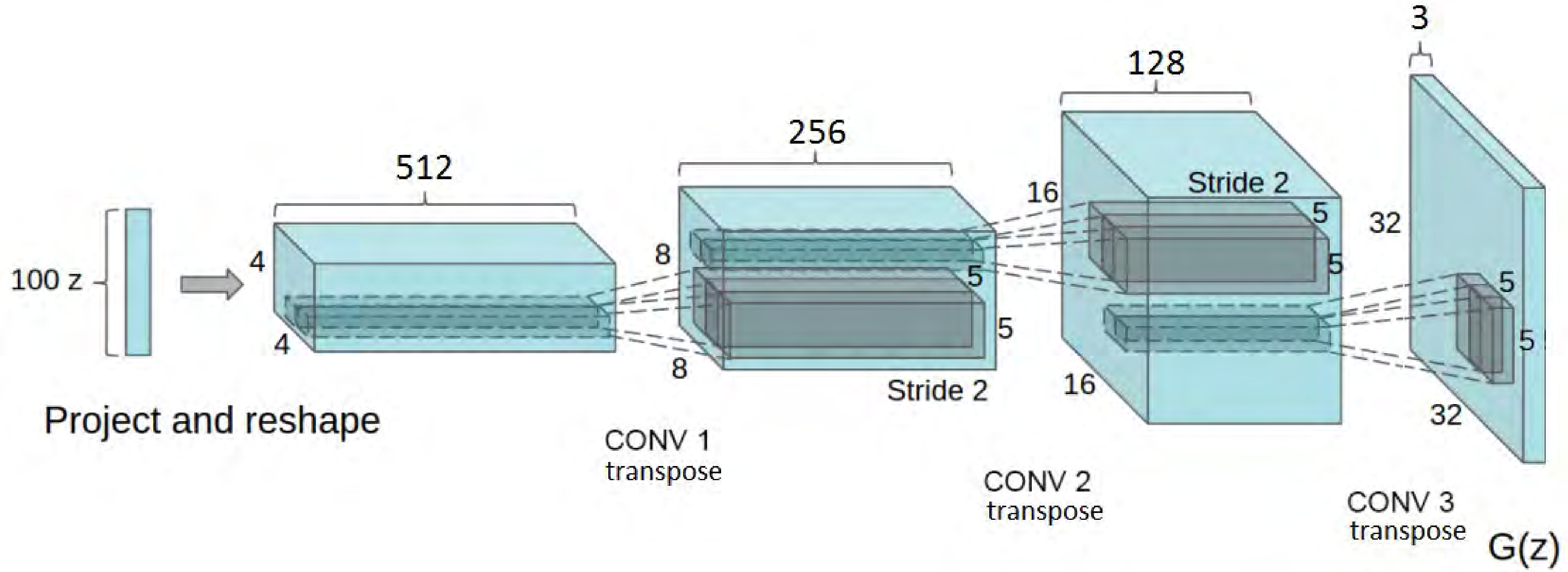
- **Generator** tries to produce data that come from some probability distribution.
- **Discriminator** acts like a judge. It gets to decide if the input comes from the generator or from the true training set.

Objective of Generator and Discriminator

- **Generator** trying to maximize the probability of making the discriminator mistakes its inputs as real.

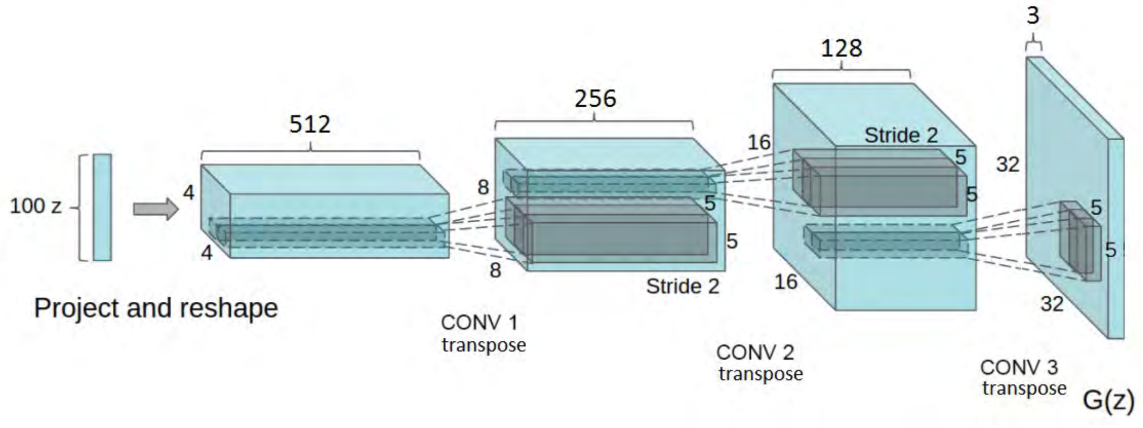
- **Discriminator** guiding the generator to produce more realistic images.

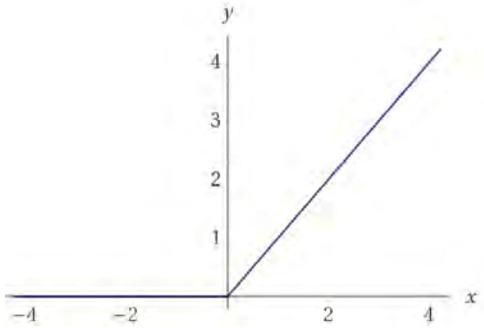
Generator



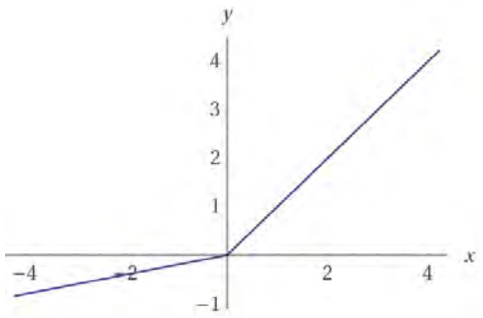
- Ideally generator would capture the general training data distribution
- There are no fully connected and pooling layers
- Generator starts with a random vector z (drawn from a normal distribution).

Discriminator





RELU



Leaky RELU

- 4 layer CNN with batch normalization
  - (except its input layer) and leaky ReLU activations
- There are no fully connected and pooling layers
- Generator starts with image tensor
- Discriminator outputs probabilities.
- Input to Discriminator:
  - Half of the time it receives images from the training set
  - The other half from the generator.

Losses

- Discriminator receives images from both the training set and the generator
- GANs need two optimizers:
  - One for minimizing the discriminator
  - One for minimizing generator's loss functions
- Discriminator receives two very distinct types of batches
- Trying to optimize **a different and opposing** objective function, or loss function

Applications

- Training semi-supervised classifiers
- Generating high resolution images from low resolution counterparts.

Training Tips

- When training the discriminator, hold the generator values constant; and vice versa;
  - i.e. train against a static adversary. For example, this gives the generator a better read on the gradient it must learn by.

- Pretraining the discriminator against ground truth before you start training the generator will establish a clearer gradient.

- Each side of the GAN can overpower the other.

- GANs take a long time to train. GPUs adviseable

Fake Celebrities

https://www.theverge.com/2017/10/30/16569402/ai-generate-fake-faces-celebs-nvidia-gan

While watching video think about

- **Opportunities**
- **Limitations**

Using Deep Learning for Image-Based Plant Disease Detection

Sharada Prasanna Mohanty[1,2], **David Hughes**[3,4,5], **and Marcel Salathé**[1,2,6]

- **Motivation :** Crop diseases remain a major threat to food supply worldwide.

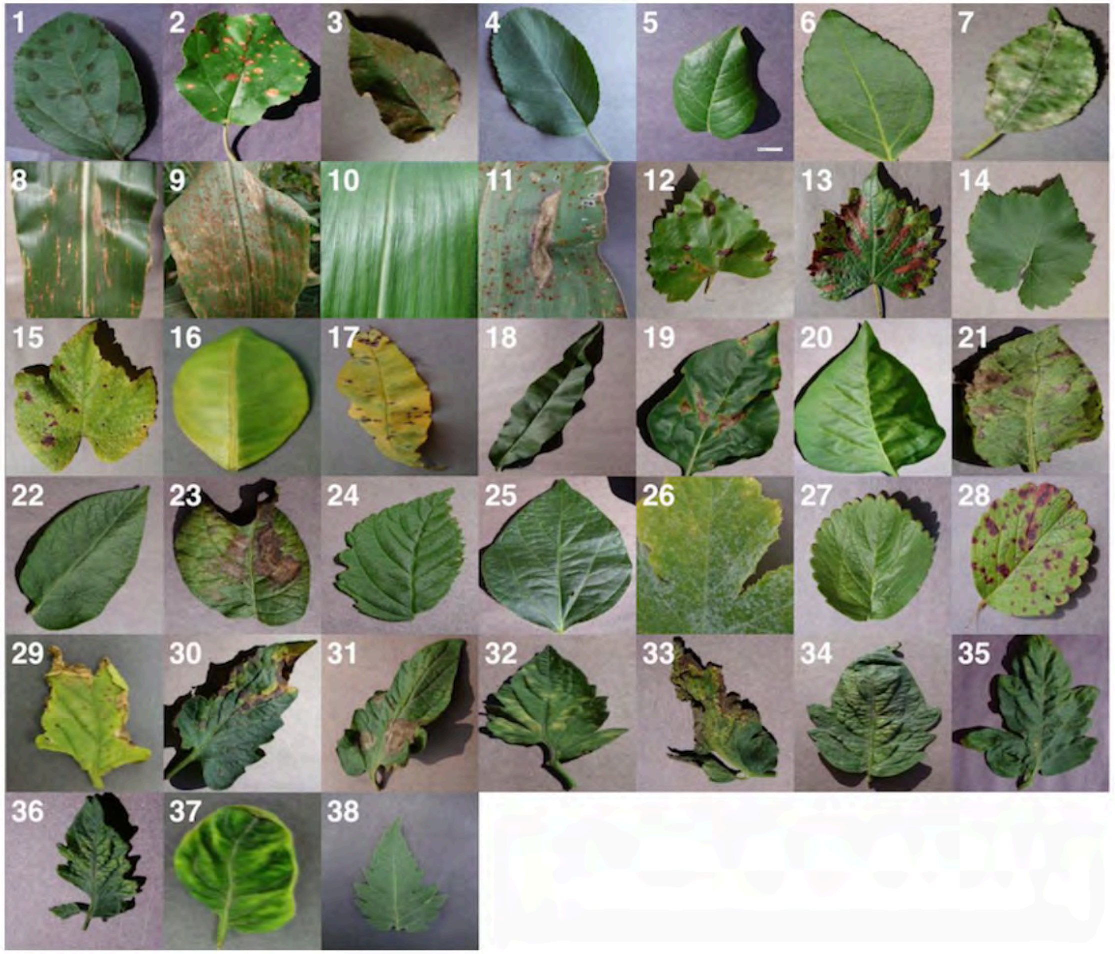
- **Objective :** Deep learning approach to enable automatic disease diagnosis through image recognition of
  - Diseased and
  - Healthy plant leaves, a deep convolutional

Step 1: Data Collection

AI project is only as smart as its garbage training set

- Collect data yourself
- Leverage internal data
- Public Dataset

Plant Village



1) Apple Scab, Venturia
2) Apple Black Rot
3) Apple Cedar Rust

8) Corn Gray Leaf Spot
9) Corn Common Rust
10) Corn healthy
11) Corn Northern Leaf Blight

Plant Village Data:

- 54,306 images
- 14 crop species
- 26 diseases

How to build model to predict disease and crop image of leaf taken using a phone?

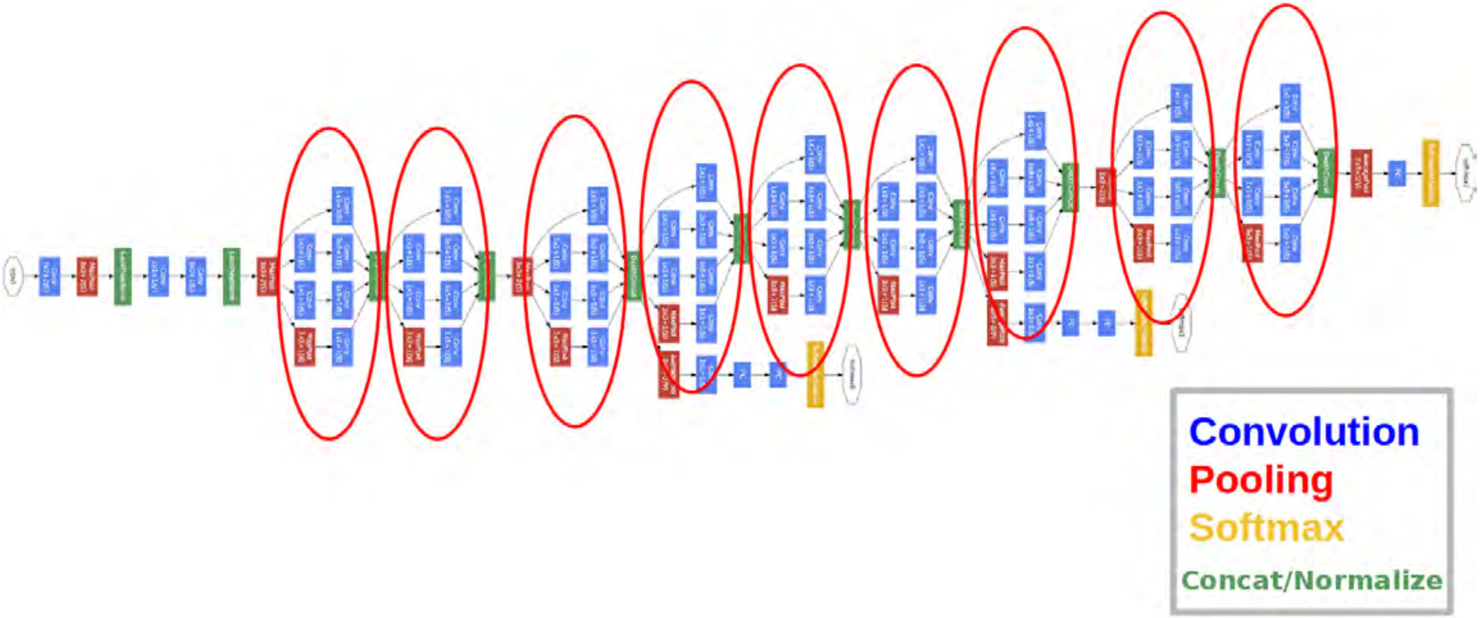
SOCIETY OF
ACTUARIES®

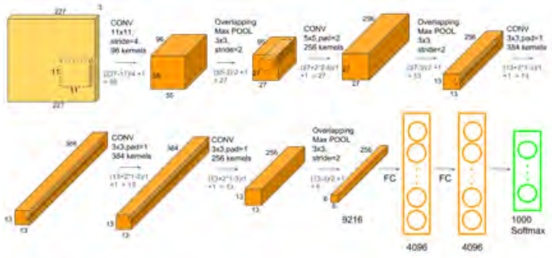
Pre-processing of data

(a) **Leaf 1**: Color (b) **Leaf 1**: Grayscale (c) **Leaf 1**: Segmented

(d) **Leaf 2**: Color (e) **Leaf 2**: Grayscale (f) **Leaf 2**: Segmented

Model Selection

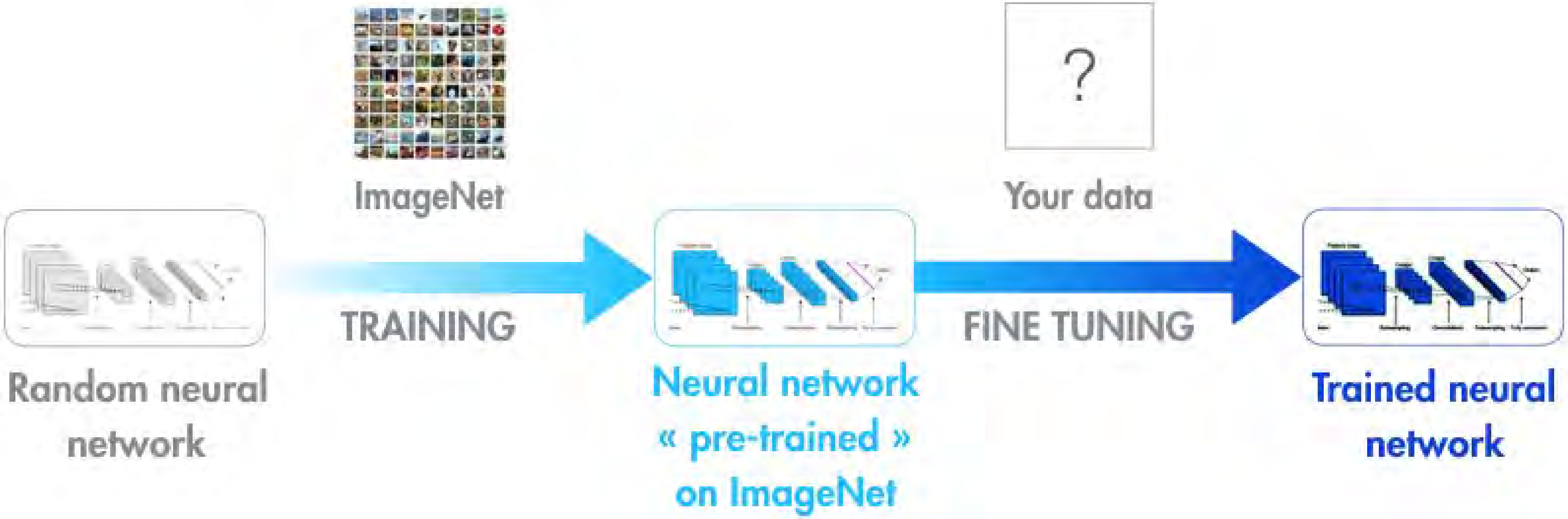




Transfer Learning

- Transfer Learning is the reuse of a pre-trained model on a new problem.
- Can train Deep Neural Networks with comparatively little data.

Transfer Learning



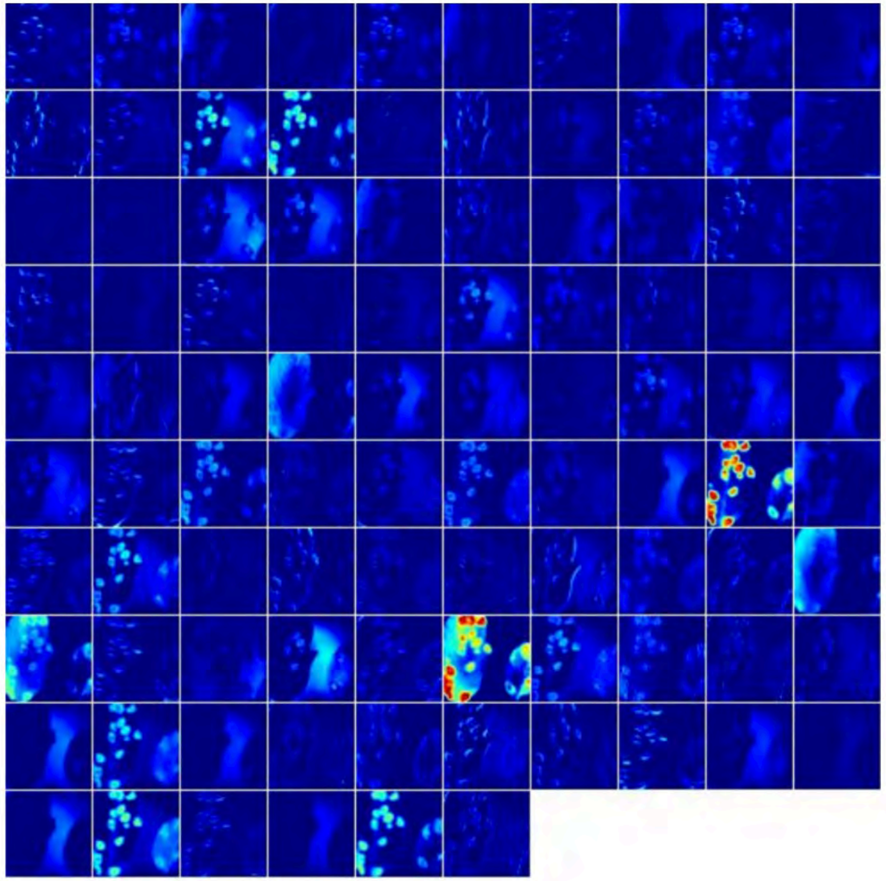
Source: https://medium.com/owkin/transfer-learning-and-the-rise-of-collaborative-artificial-intelligence

ImageNet Challenge

IMAGENET

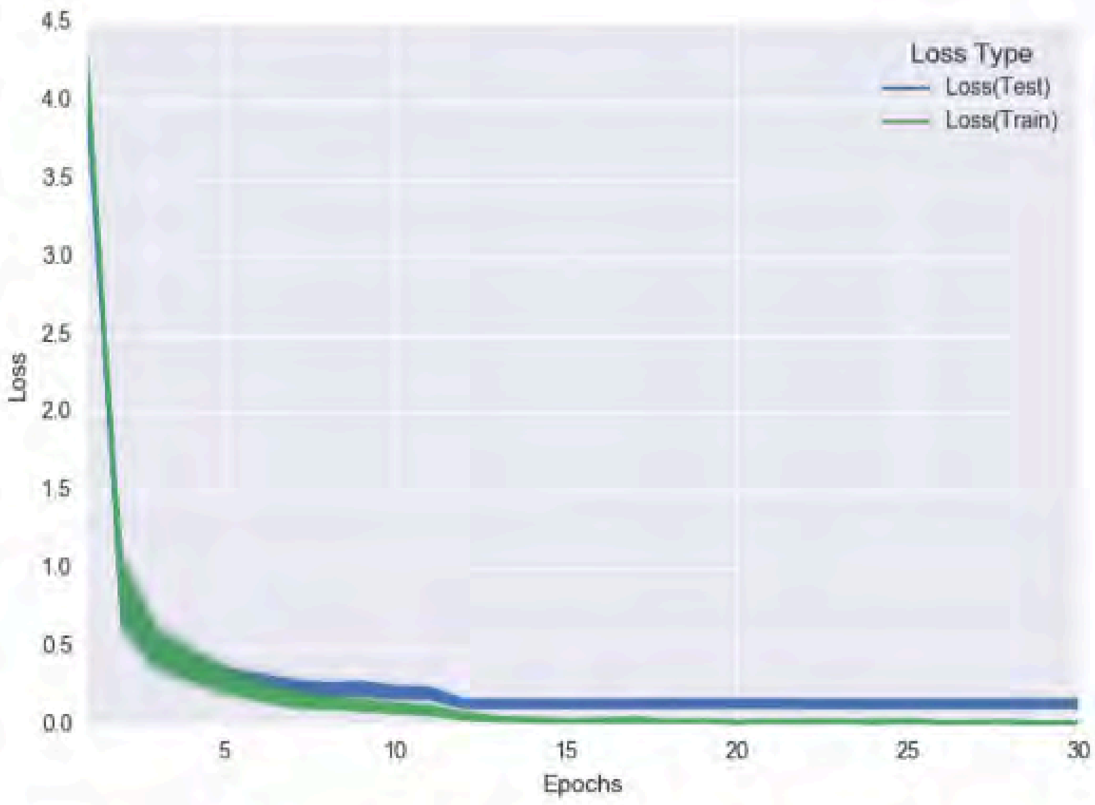
- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Interpreting DL Models



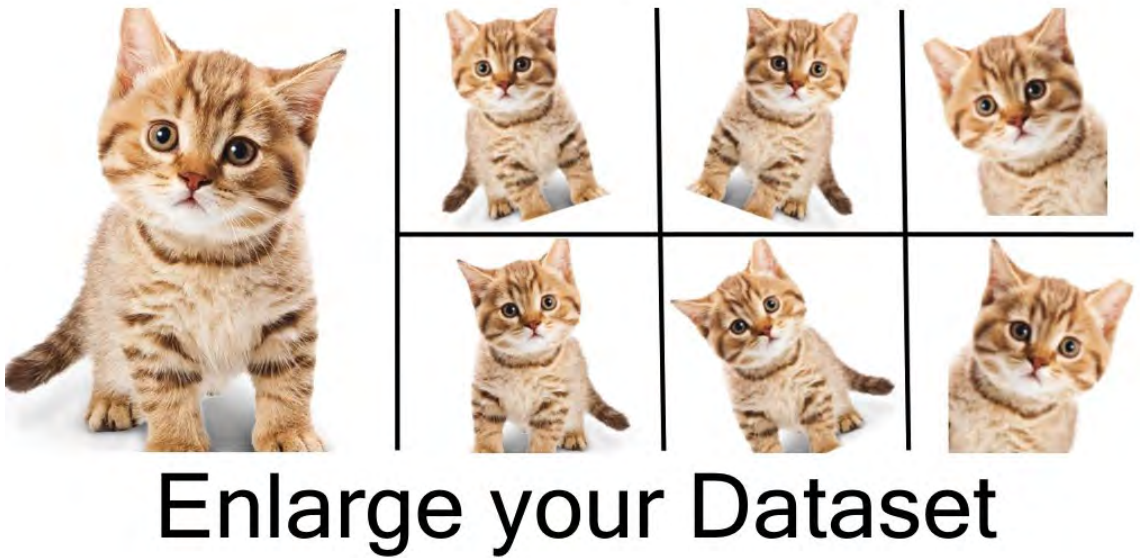
(b) Visualization of activations in the first convolution layer(*conv1*) of an AlexNet architecture trained using **AlexNet:Color:TrainFromScratch:80-20** when doing a forward pass

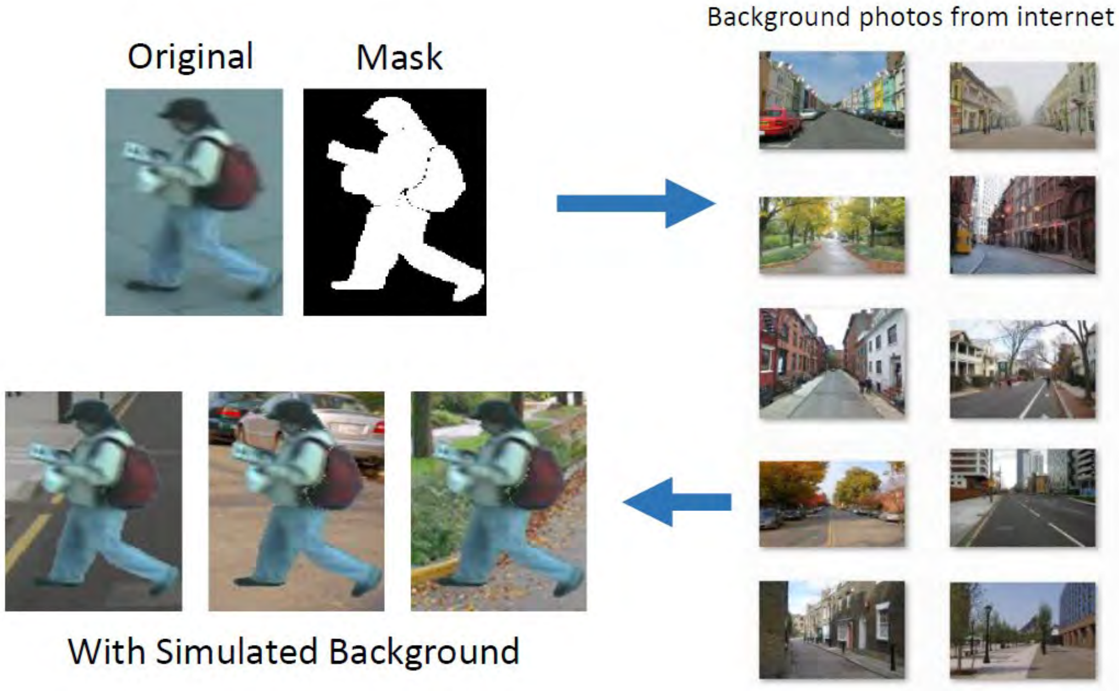
How long to train model?



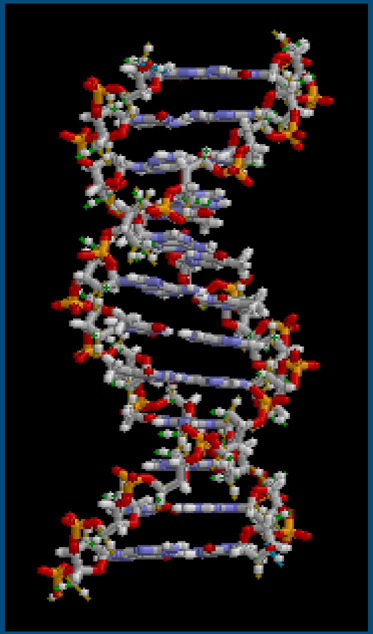
(c) Comparison of progression of train-loss and test-loss across all experiments.

Additional Tuning





2019 PREDICTIVE ANALYTICS SYMPOSIUM



Session 14: AP - Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) - Moving Beyond Basic Neural Networks for Innovative Advantages

19-September-2019

David Snell, FALU, FLMI, ASA, MAAA, CLU, ChFC, ARA, ACS, MCP dave@ActuariesAndTechnology.com

Balathasan Giritharan Ph. D giritharanb@gmail.com

SOCIETY OF ACTUARIES®

ANN common terms

- ***Node*** – this is becoming the more common term for what used to be called a neuron. A node is a more apt term since it is a lot less complex than a biological neuron.
  - Input node – contains an input to the ANN; always numeric; usually standardized or mapped to a value from 0 to 1, or -1 to 1. Each input node represents a single dimension (feature). All of the features are stored in a vector called the input layer.
  - Bias node – an extra node which is added to the input layer and to each hidden layer. It usually has a value of 1 or -1. This enables the activation function to be shifted left or right. It also helps in the training process. The output layer never has a bias node. The bias does not change, but it is multiplied by its edge weight so the input to the next layer is essentially determined by the weight.
- ***Layer*** – a collection (vector) of nodes (neurons)
  - Input layer – a vector of features (input nodes) that travels through the ANN via two functions: a summation function and an activation function.
  - Output layer – the output, or result (target) of the ANN
  - Hidden layer(s) – intermediate vectors of nodes between the input layer and the output layer. The more hidden layers in the ANN, the more it is associated with Deep Learning. Correspondingly, the 'deeper' the ANN, the more difficult it can be to explain the processing involved. The choice of the number of hidden layers and the number of nodes in each layer is still part science and part art. Expert practitioners learn to vary these to fit specific needs.

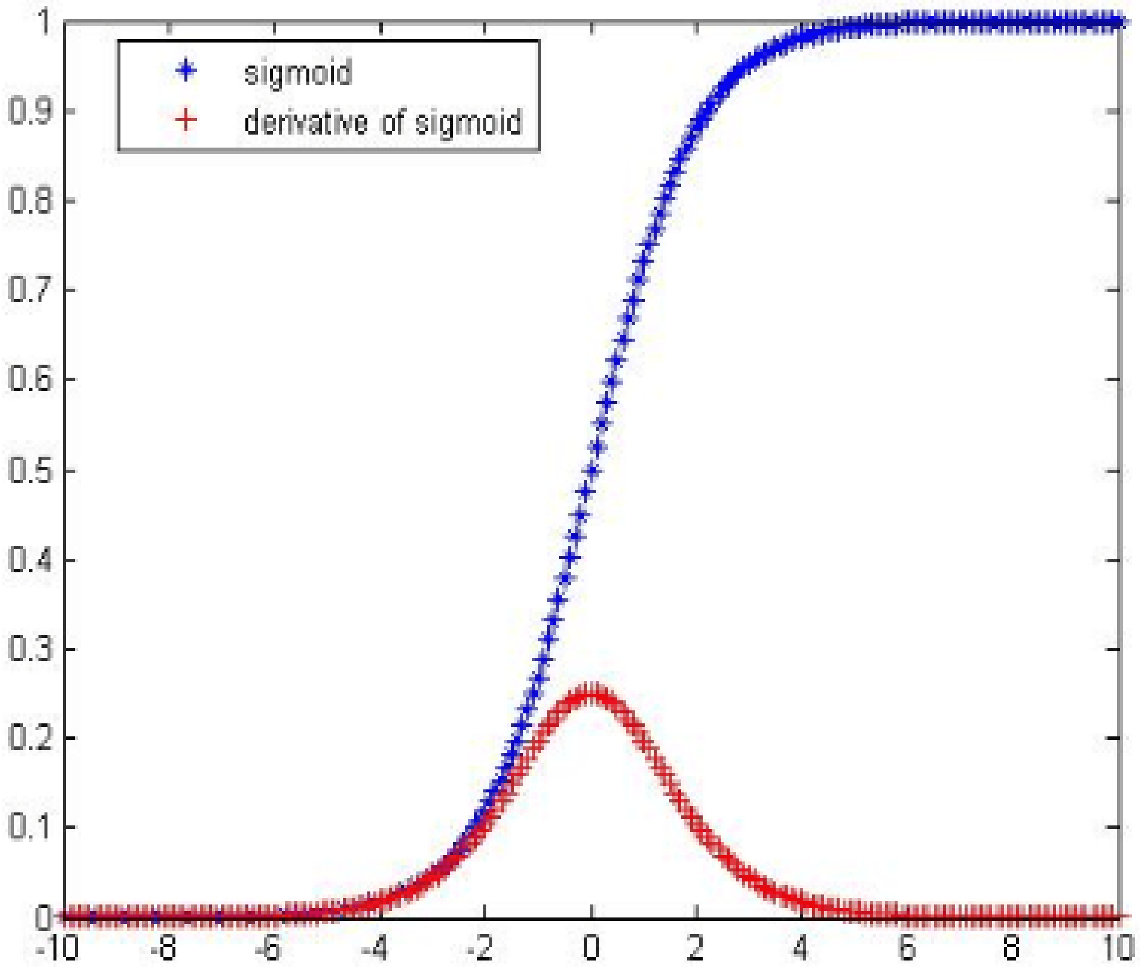
ANN common terms (continued)

- ***Edge*** – this is the connector between nodes. It is the ANN counterpart of the biological synaptic cleft. It forms the link between a node in layer n to a node in layer n+1. It's main importance is to carry a weight.
- ***Weight*** – instead of all the calcium and sodium ions and internal resistances to communication between biological neurons, a weight is just a modifier to the signal from one ANN node to another ANN node. The weights are what get modified during the training process. In fact, they are the main 'tuning knob' used in network training.
- ***Hyperparameters*** – tuning knobs to help train a network. Unlike the weights, hyperparameters are set ahead of time, outside of the ANN. Hyperparameters include the number of layers, the number of nodes in each layer, bias values, a learning rate schedule, momentum, and the batch size used for training (each observation, all observations, mini-batch of a portion of all observations).

ANN common terms (continued)

- ***Learning rate*** – a value that speeds up (or slows down) the algorithmic learning process. This is the size of the step taken when moving towards a global minimization of the cost function. It can be fixed (static) or programmed to scale back as the ANN error rate approaches a minimum.
- ***Momentum*** – a value (predetermined) used to help avoid local rather than global minimization by pushing an ANN out of a local minimum. This is normally done during the backpropagation process.
- ***Activation function*** – this is a function that maps the total net input to a node and then determines the output from that node (again, multiplied by the edge weight applied). Common activation functions include the sigmoid (logistic), ReLU, hyperbolic tangent, softmax, and step functions.
- ***ReLU function*** – an activation that has gained popularity because of its great performance in deep neural networks. All input < 0 is set to zero, and all input > 0 is passed through without alteration.
- ***Logistic function*** – this involves a sigmoid curve and maps the input to a value between 0 and 1. Like the tanh (hyperbolic tangent) function, this can break the linearity of an ANN and enable it to solve more complex problems. The tanh function limits the output range to -1 to +1.

The derivative of a sigmoid function s(x) is s(x)(1-s(x))



$$\frac{d}{dx}s(x) = \frac{(e^{-x})}{(1+e^{-x})^2}$$

This part is not intuitive... but let's add and subtract a 1 to the numerator (this does not change the equation).

$$\frac{d}{dx}s(x) = \frac{(e^{-x}+1-1)}{(1+e^{-x})^2}$$

$$\frac{d}{dx}s(x) = \frac{(1+e^{-x}-1)}{(1+e^{-x})^2}$$

$$\frac{d}{dx}s(x) = \frac{(1+e^{-x})}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2}$$

$$= \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2}$$

$$= \frac{1}{(1+e^{-x})} - \left(\frac{1}{(1+e^{-x})}\right)\left(\frac{1}{(1+e^{-x})}\right) \text{ // factor out a } \frac{1}{(1+e^{-x})}$$

$$= \frac{1}{(1+e^{-x})}\left(1 - \frac{1}{(1+e^{-x})}\right)$$

Hmmm.... look at that! There's actually two sigmoid functions there... Recall that the sigmoid function is, $s(x) = \frac{1}{1+e^{-x}}$. Let's replace them with s(x).

$$s'(x) = \frac{d}{dx}s(x) = s(x)(1 - s(x))$$

http://kawahara.ca/how-to-compute-the-derivative-of-a-sigmoid-function-fully-worked-example/

ANN common terms (continued)

- ***Step function*** – although visually intuitive, the step function is of limited value in ANNs because it produces an output of 0 or 1 and nothing in between. This can impede the learning process.
- ***Matrices*** – these provide a fast, efficient, and less error-prone way to calculate large amounts of data transformations during both forward and backward propagation.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

- ***Cost function*** – this is also called a loss function or error function. It transforms the output of an ANN to a number which serves as a measure of how wrong the ANN is. It reflects the difference between the ANN's actual output and its target output. Common cost functions include sum of squares error, mean squared error, root mean squared error.

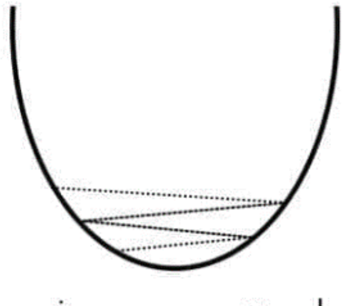
- ***Mean squared error*** – squaring the errors keeps positive and negative errors terms from cancelling each other out. Also, squaring helps the ANN converge faster because the larger errors are emphasized and the derivatives result in larger steps toward the global minimum. Taking the average (mean) permits meaningful comparisons between groups of varying sizes.
- ***Gradient*** – the gradient is a partial derivative. The gradient is used to determine the amount and the direction of adjustments to weights. The gradients for the output layer are always calculated first, since these values are used to calculate the previous layer partial derivatives.
- ***Back propagation*** – the process of computing gradients from the output layer back through the ANN to the input layer, adjusting the weights along the way.

$$\frac{\partial E}{\partial W_1} = \overset{❶}{\frac{\partial E}{\partial net_{c1}}} \; \overset{❷}{\frac{\partial net_{c1}}{\partial out_{b1}}} \; \overset{❸}{\frac{\partial out_{b1}}{\partial net_{b1}}} \; \overset{❹}{\frac{\partial net_{b1}}{\partial W_1}}$$

back propagation uses the chain rule

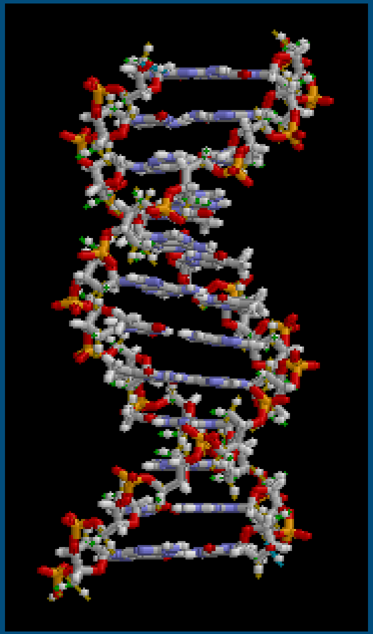
- *Stochastic Gradient Descent* (also called **SGD**) With SGD, the weights in a network are modified after every training set element. With SGD, it is important to note that the training set is shuffled because the order of the data can cause the network to become biased. There are various approaches to shuffling, such as a single shuffle at the beginning of training or after every epoch. SGD is typically faster than full-batch training, especially early on in the training process. However, it can also produce much more noise, which causes the network to bounce around near the global minimum but never reach it.

  The idea behind SGD is to minimize the error of every single weight in the ANN, thus minimizing the total error of the network.



Diverging can occur when the learning rate is too large.

2019 PREDICTIVE ANALYTICS SYMPOSIUM



Session 14: AP - Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) - Moving Beyond Basic Neural Networks for Innovative Advantages

19-September-2019

David Snell, FALU, FLMI, ASA, MAAA, CLU, ChFC, ARA, ACS, MCP dave@ActuariesAndTechnology.com

Balathasan Giritharan Ph. D giritharanb@gmail.com

SOCIETY OF ACTUARIES®