Enhancing Accuracy and Reliability with Influencer Functions

Brandon Panos

Visual Inspection for Quality Control and Traceability,

September 9, 2024

Fachhochschule Nordwestschweiz

FHGR Fachhochschule Graubünden University of Applied Sciences

SWISS PHOTONICS

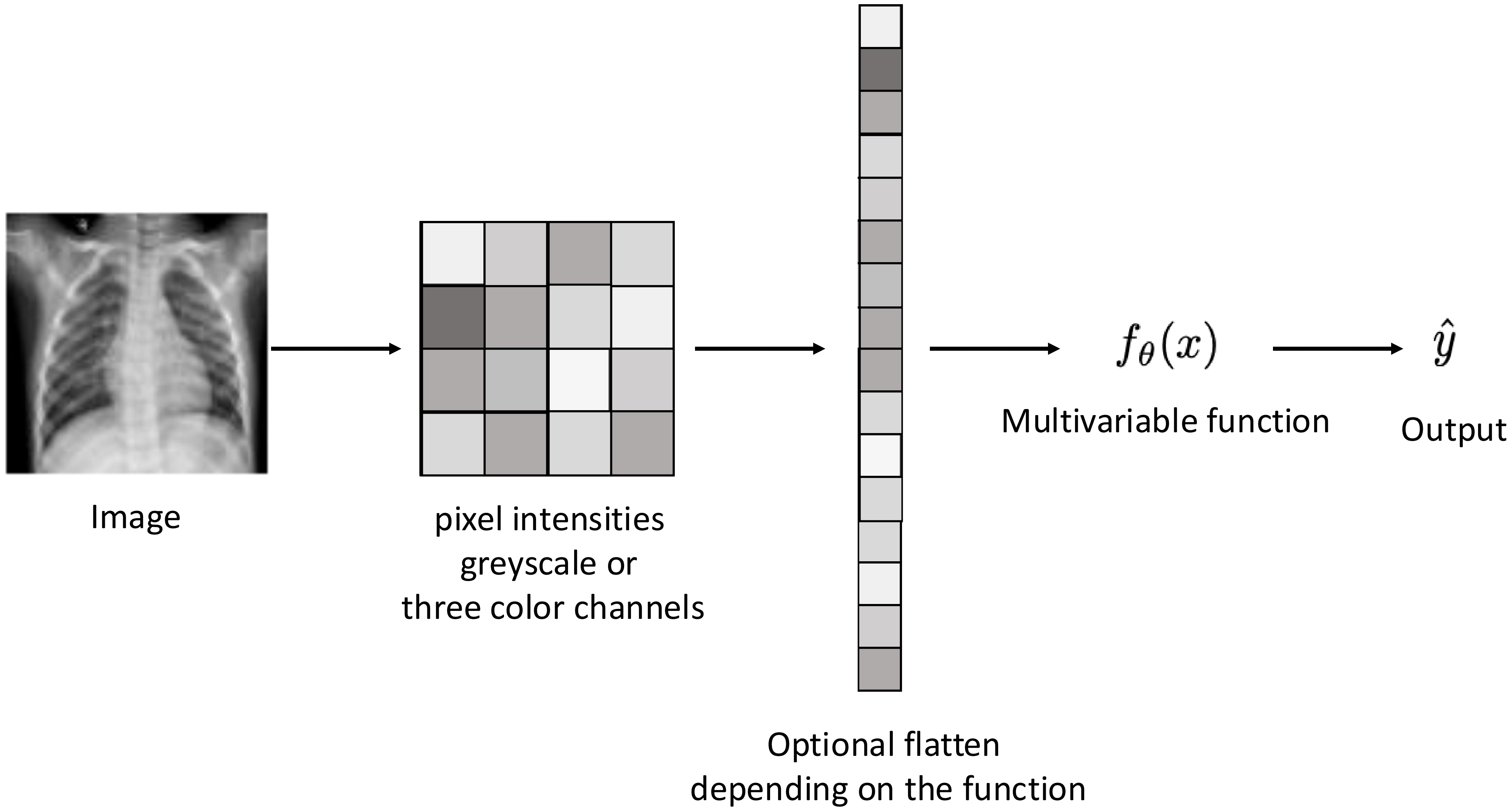
PHOTONICS INNOVATION BOOSTER

Innovation Booster

https://brandonlpanos.herokuapp.com/

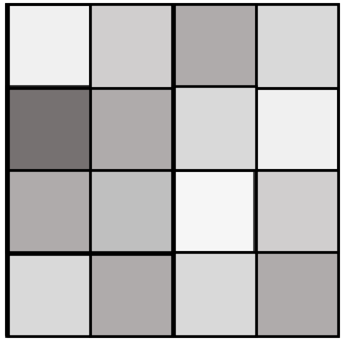
brandon.panos@fhnw.ch

The life cycle of an image: Data → Function → Output

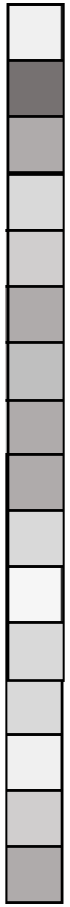


The life cycle of an image: Function can be parameterized by a NN

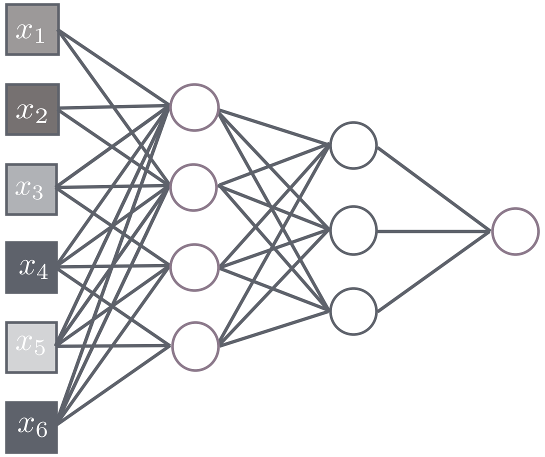
Image



pixel intensities
greyscale or
three color channels



Optional flatten
depending on the function



$$f_\theta(x)$$

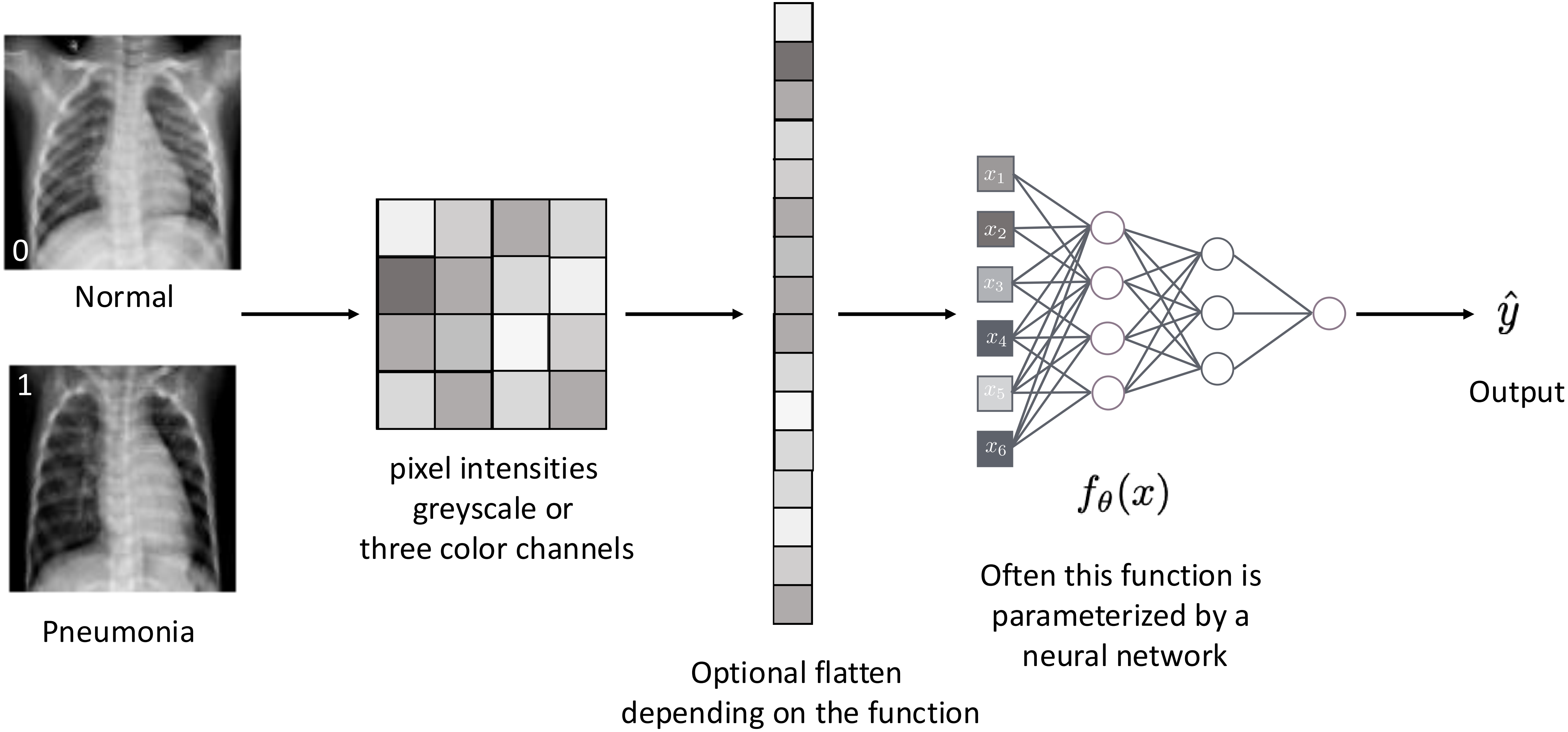
Often this function is
parameterized by a
neural network

$\hat{y}$

Output

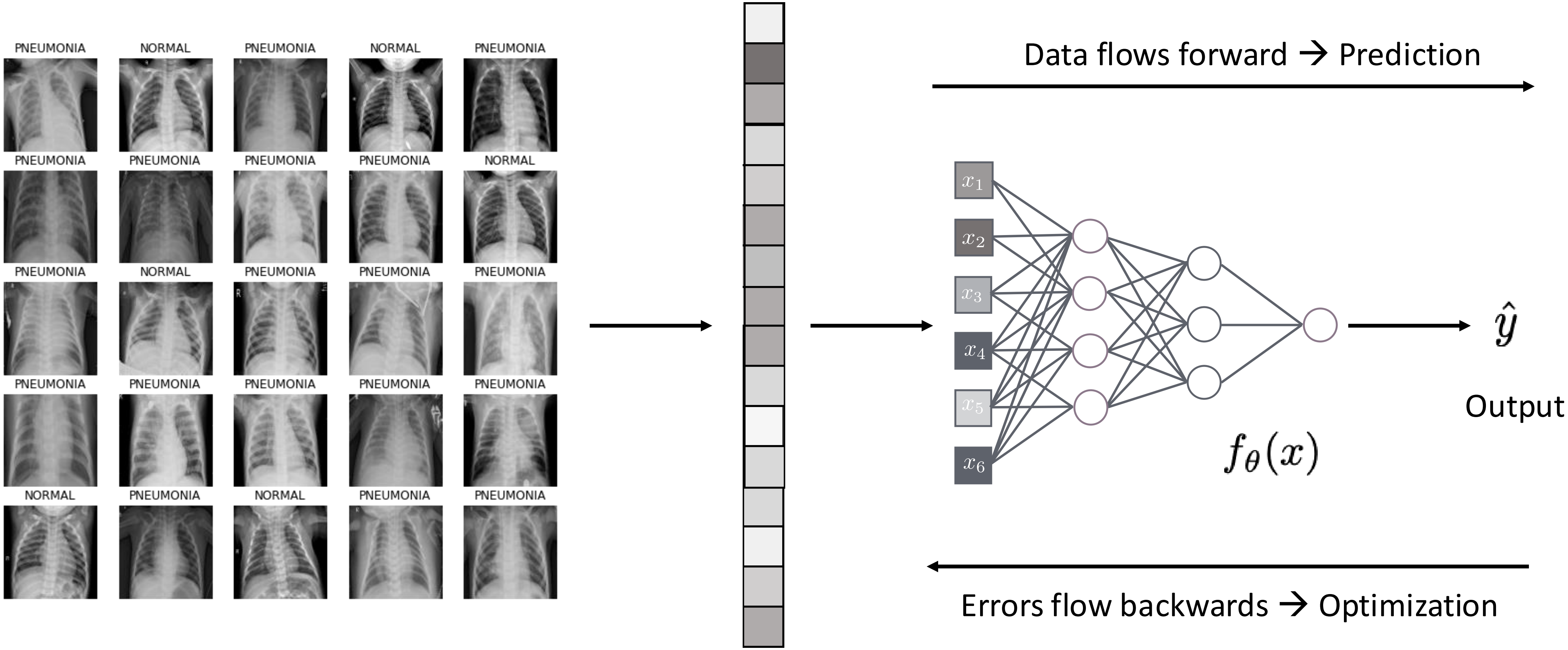
Tunable function

The life cycle of an image: Task could be classification



Binary classification

The life cycle of an image: Data driven

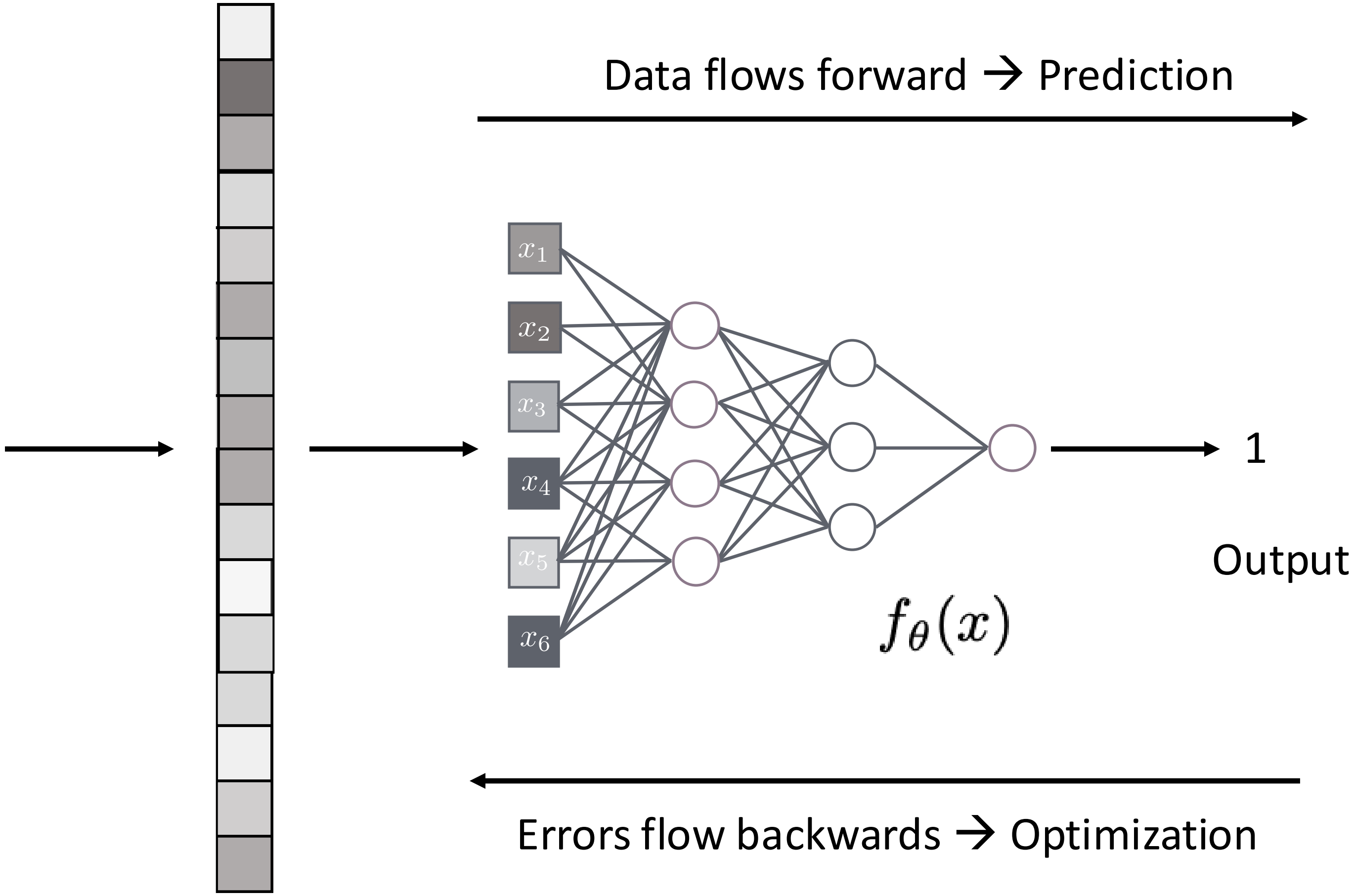


Optional flatten
depending on the function

Data driven automatic optimization

The life cycle of an image: Data driven

Pneumonia



Optional flatten
depending on the function

Scores 100 % accuracy

End of the story…..?

The life cycle of an image

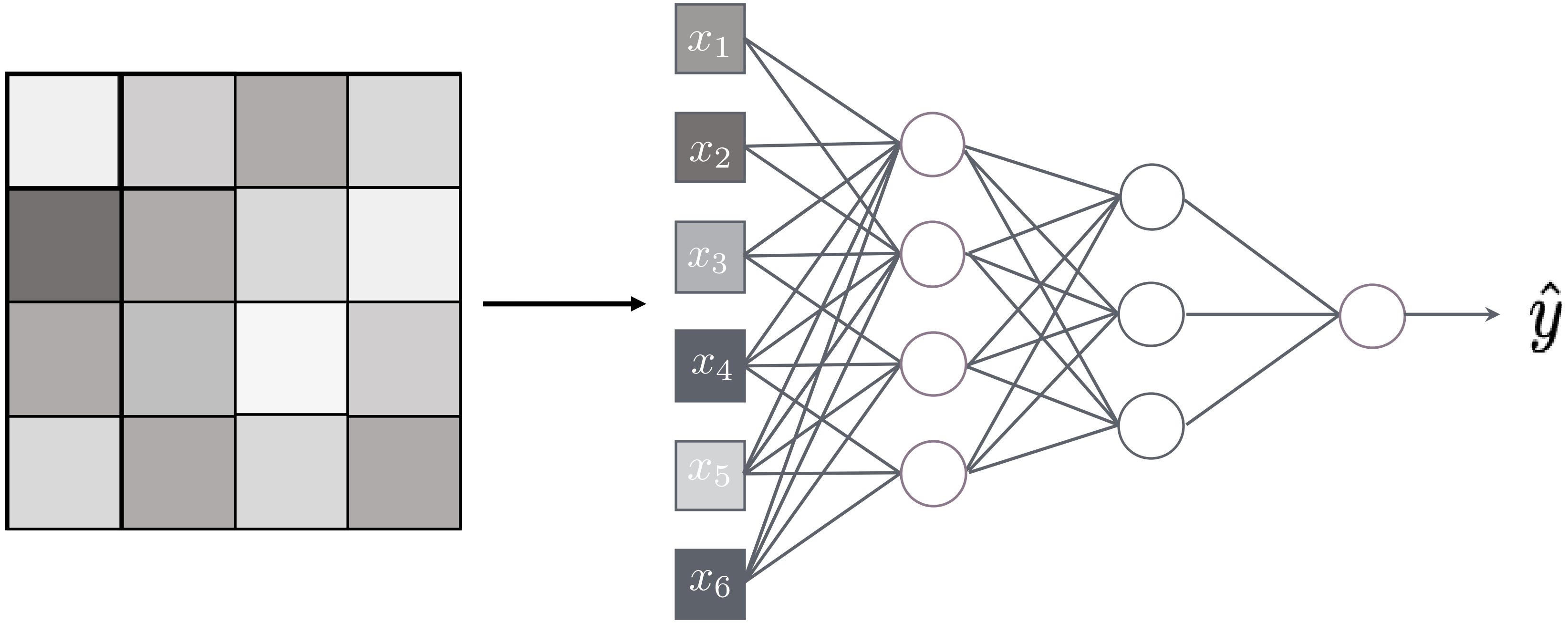
NO!

The life cycle of an image

We have no idea how the function makes its decisions

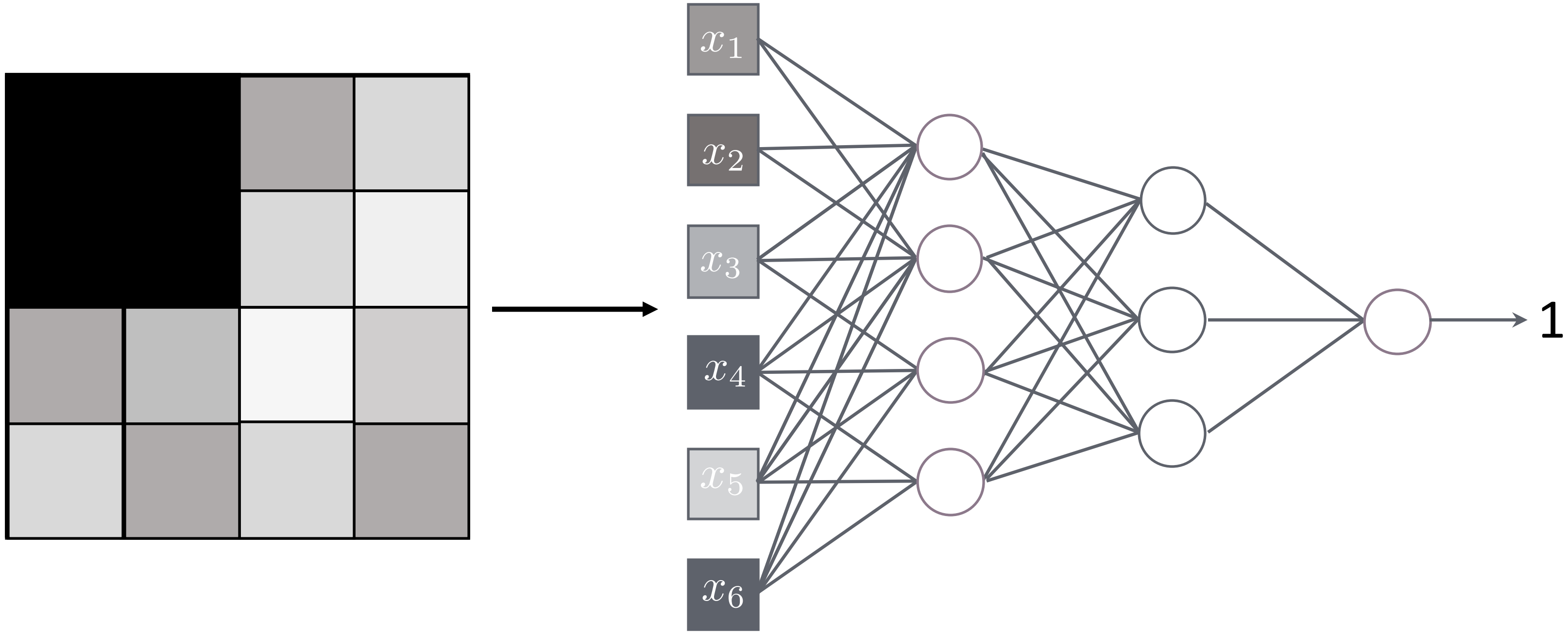
This is dangerous in fields that effect the public

Model Explainability



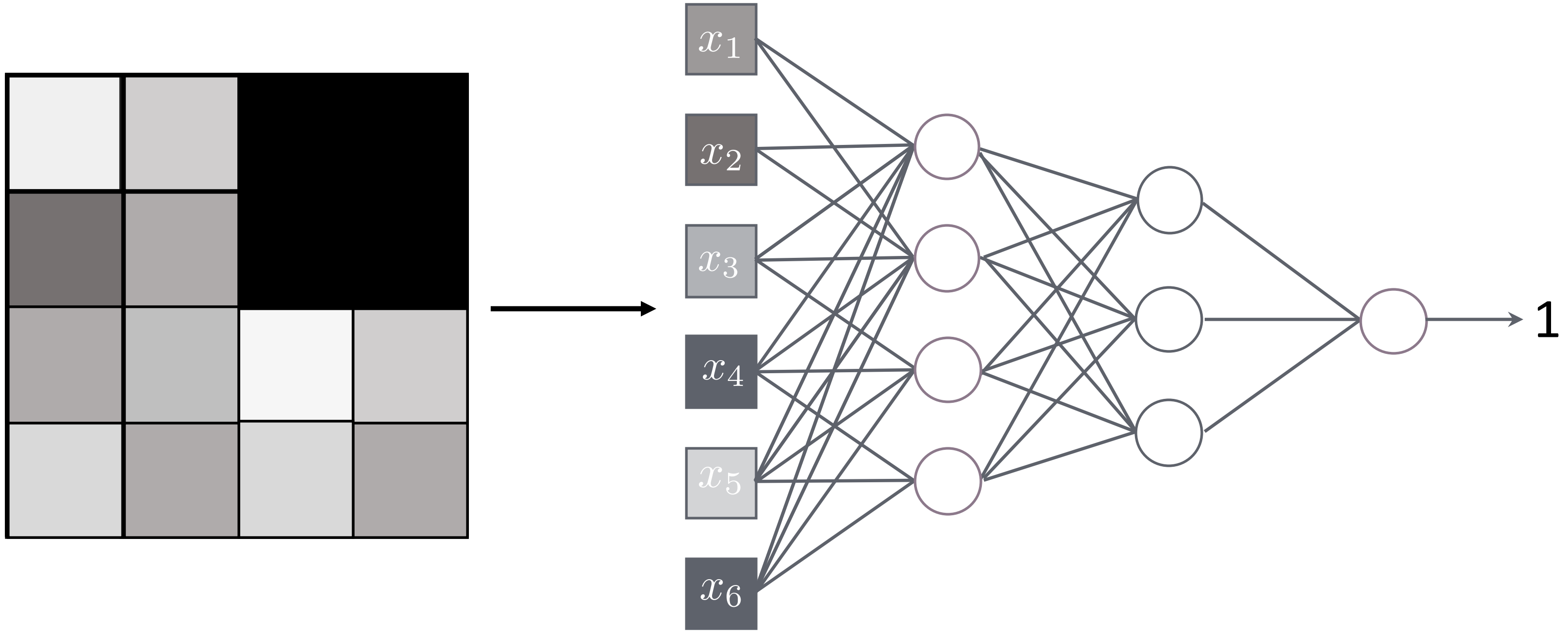
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Occlusion



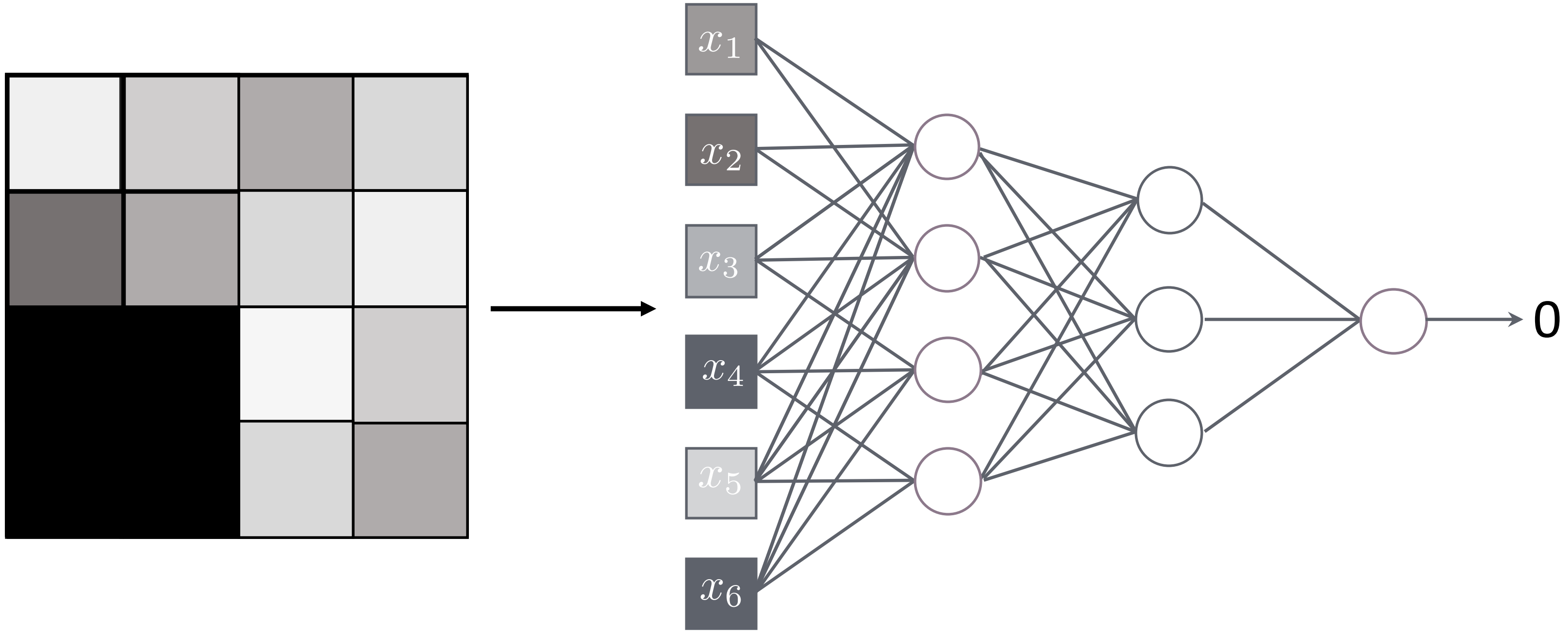
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Occlusion



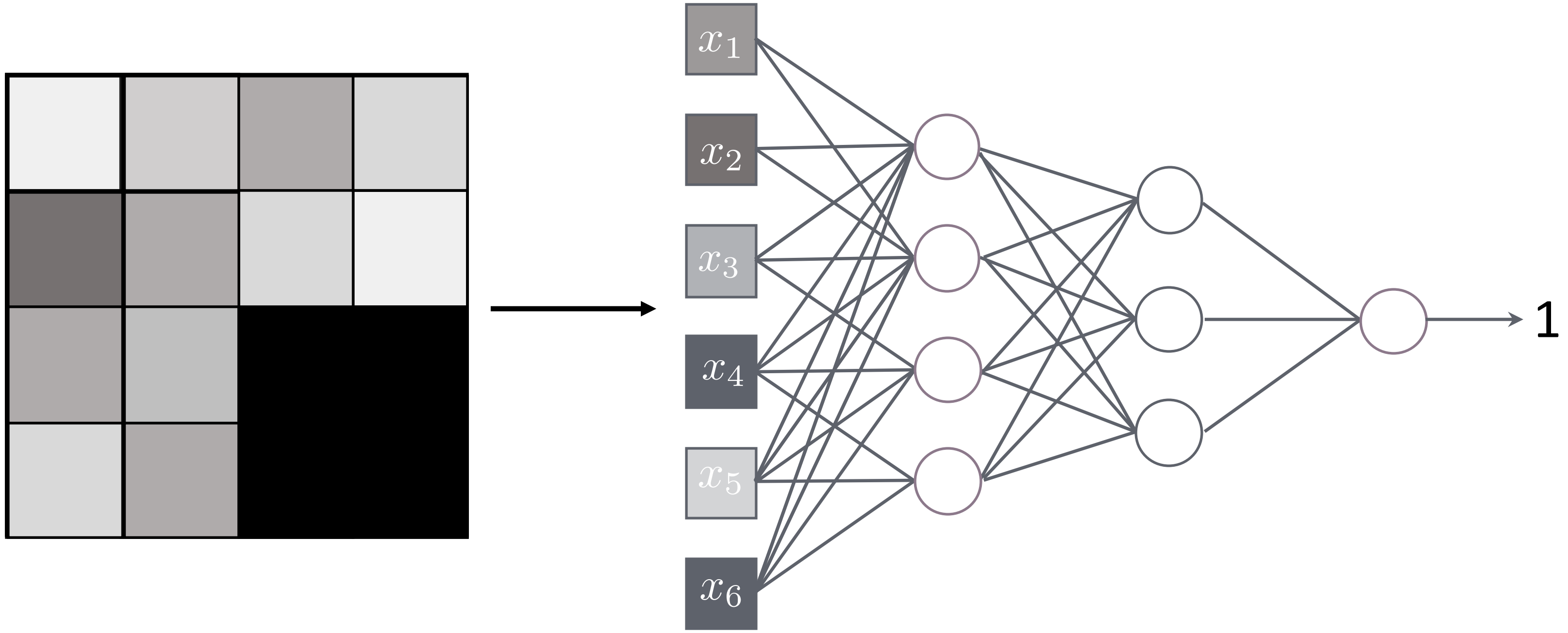
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Occlusion



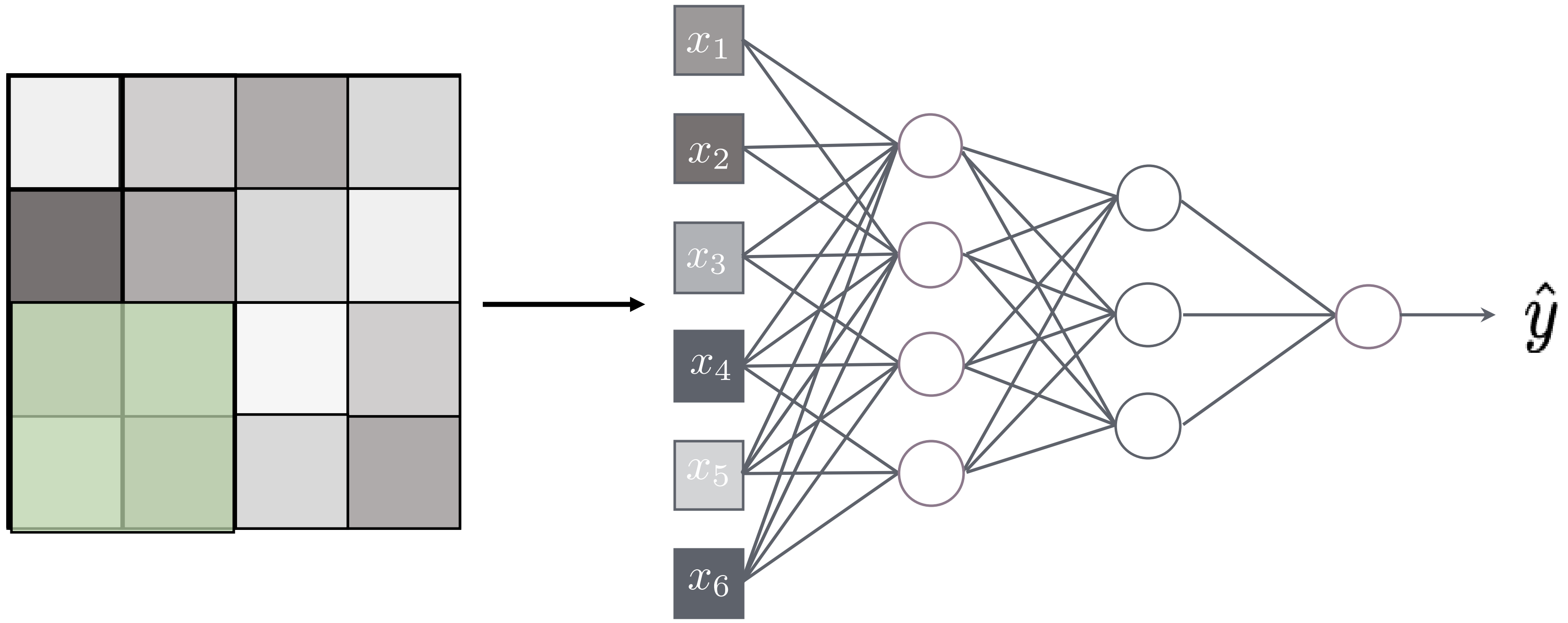
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Occlusion



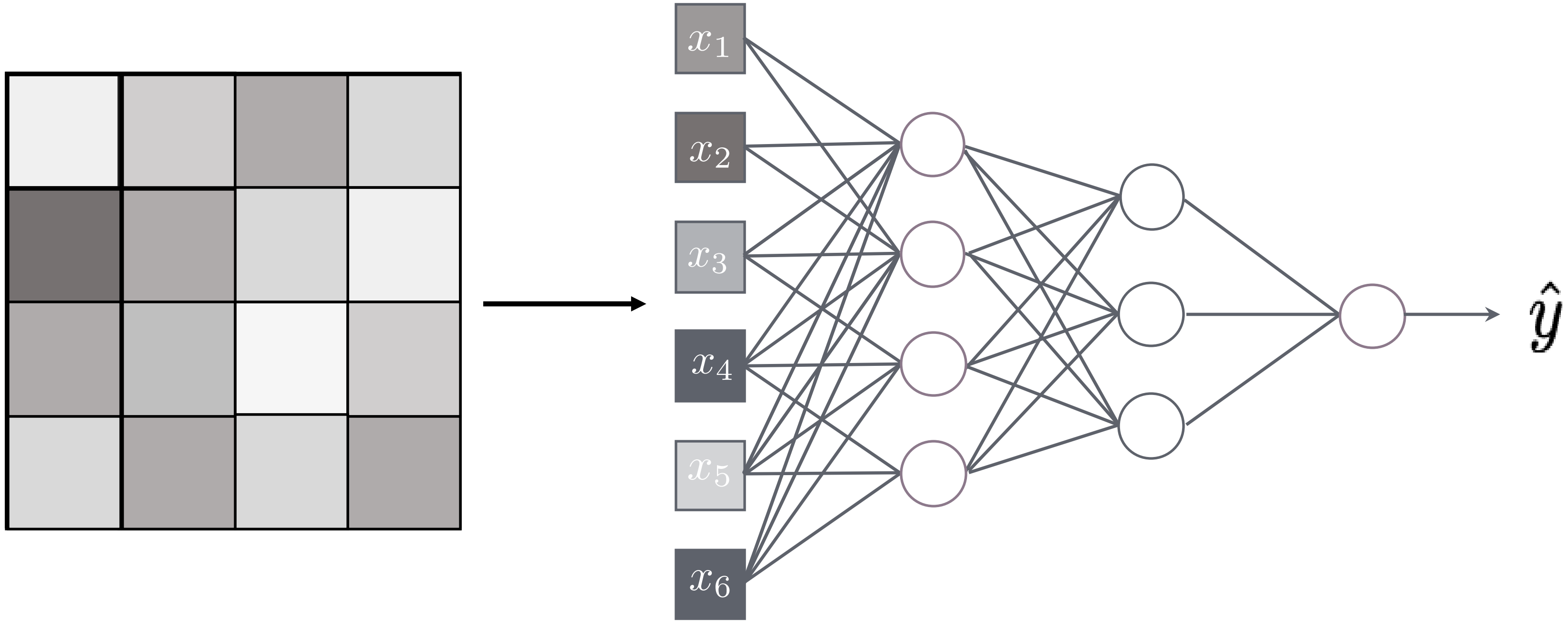
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Occlusion



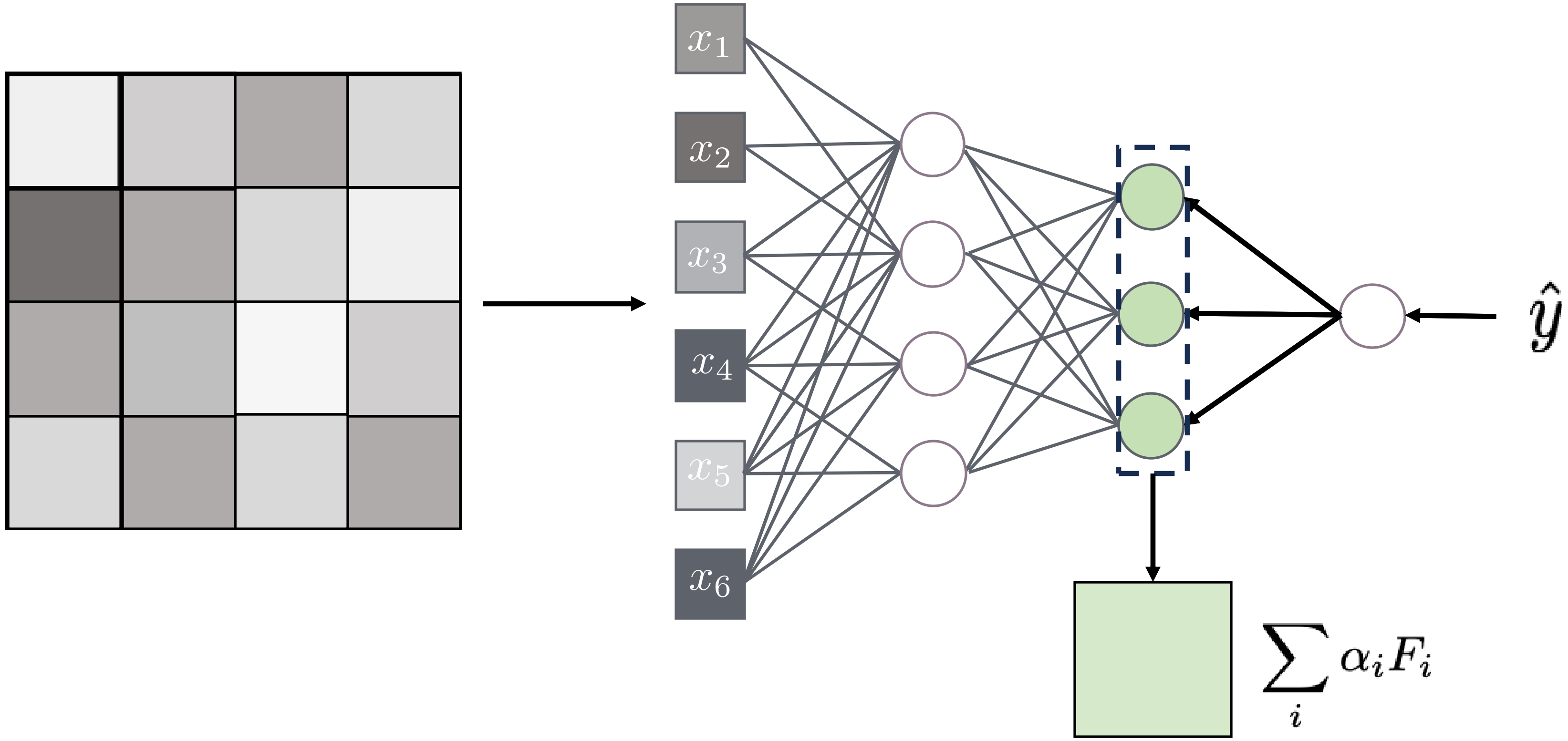
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Grad-CAM



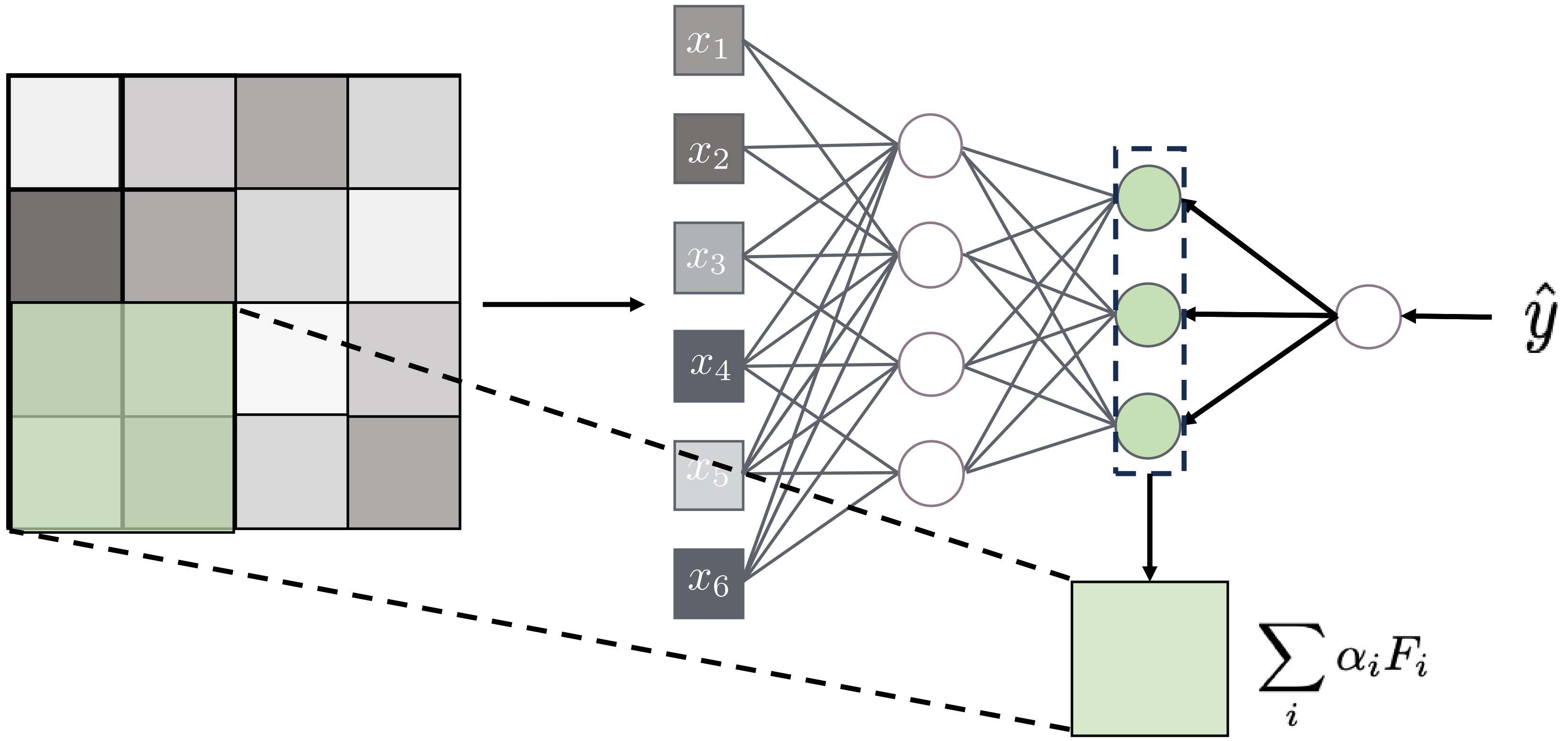
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Grad-CAM



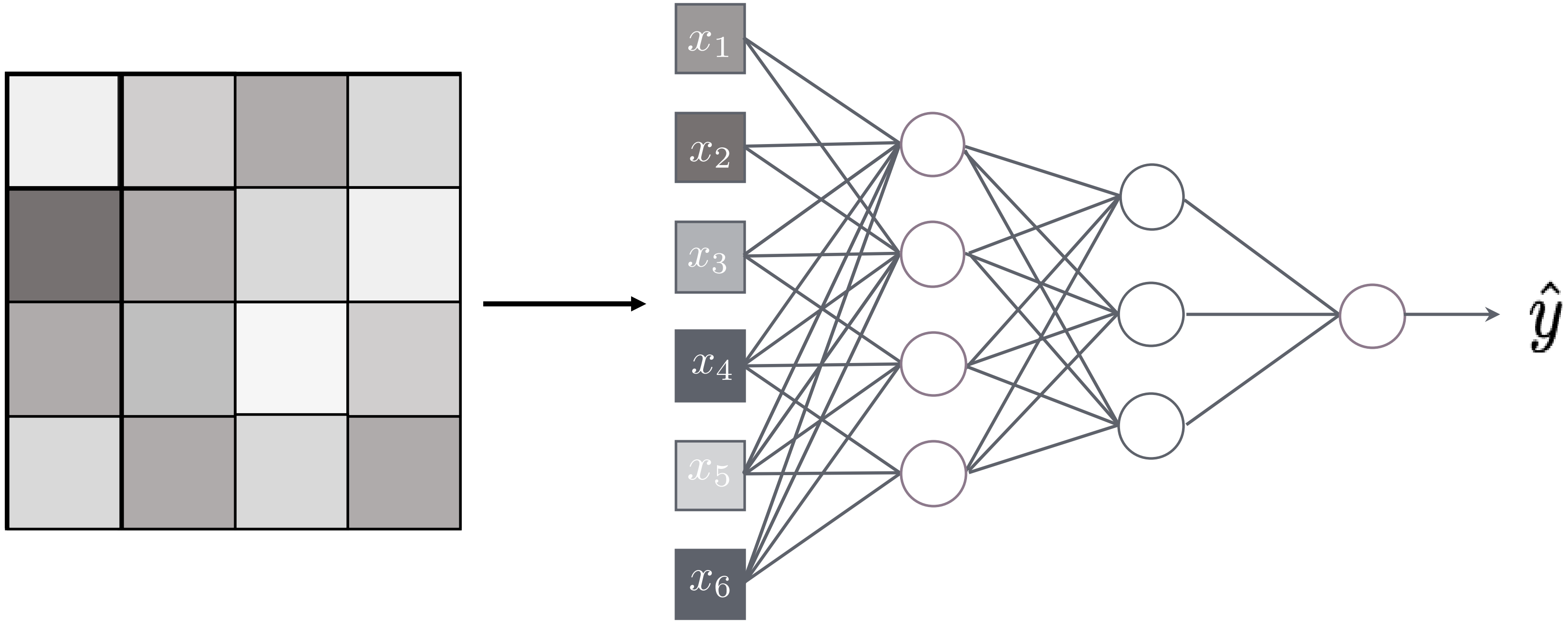
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Grad-CAM



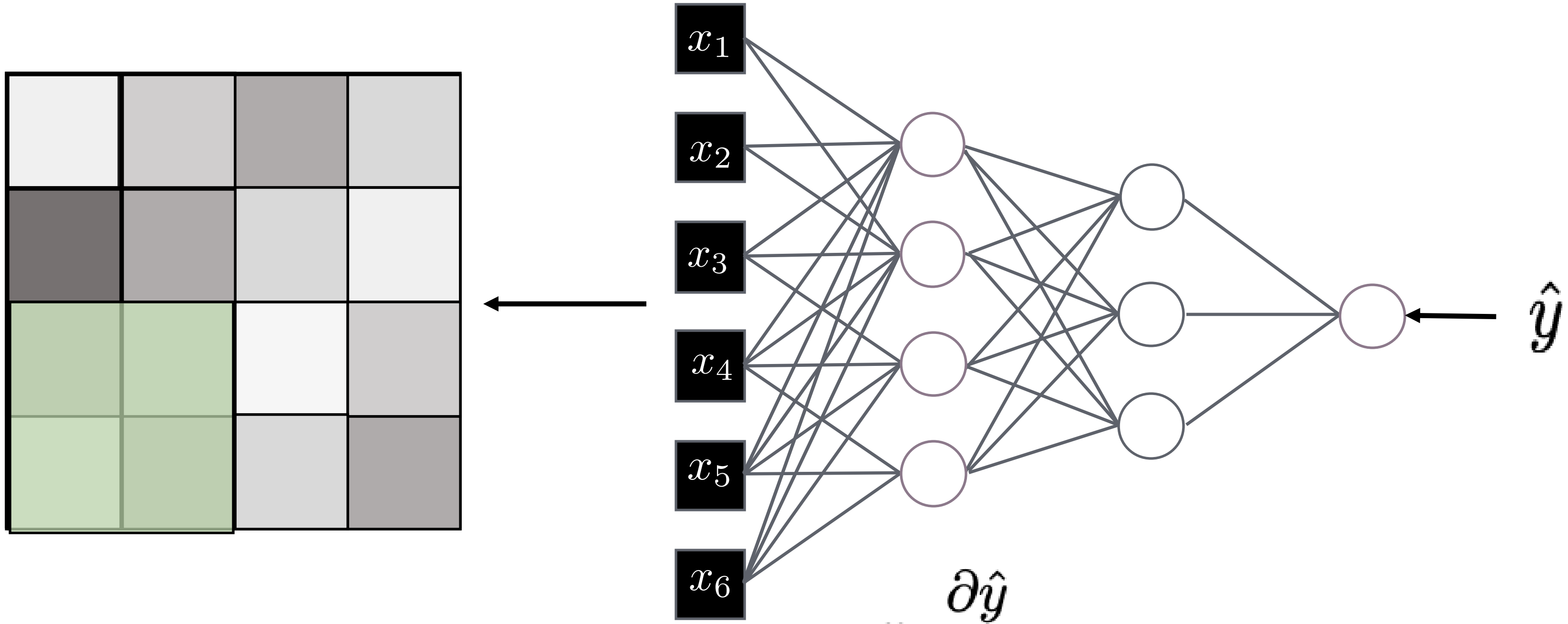
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Grad X Input



There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

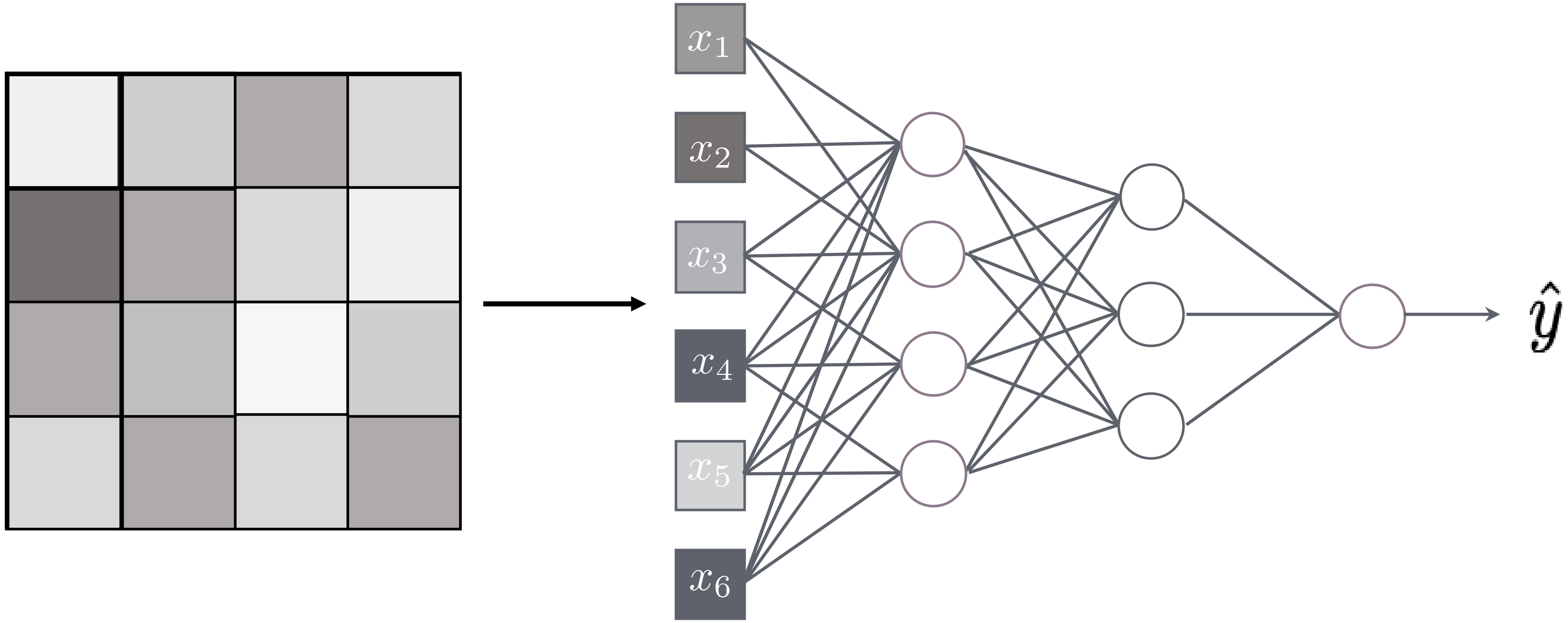
Model Explainability: Grad X Input



$$x_i \frac{\partial \hat{y}}{\partial x_i}$$

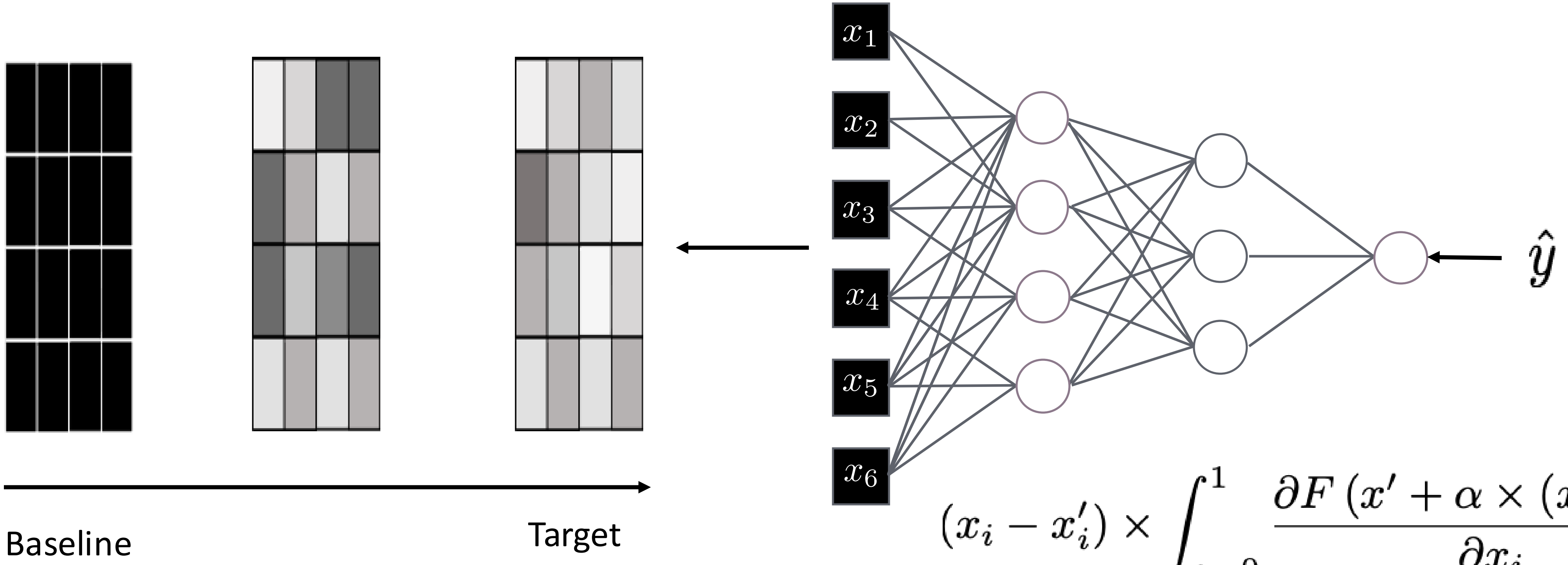
There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability: Integrated Gradients



There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

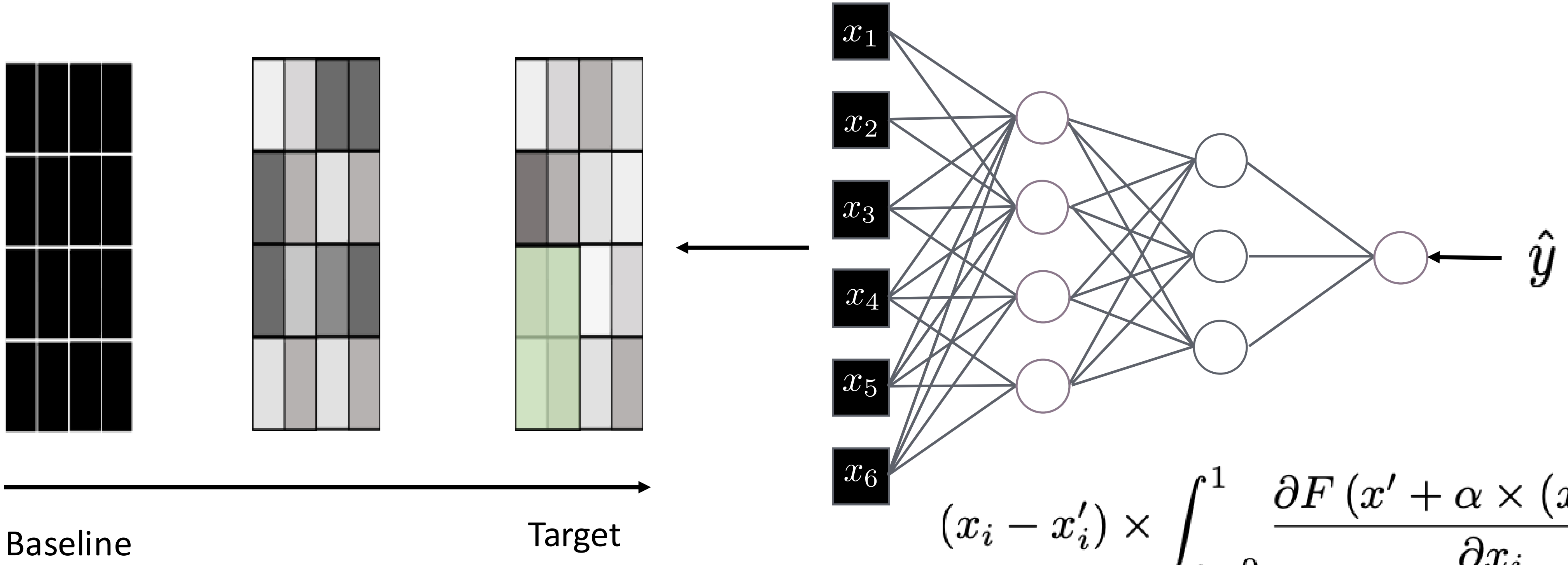
Model Explainability: Integrated Gradients



$$(x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial x_i} d\alpha$$

There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

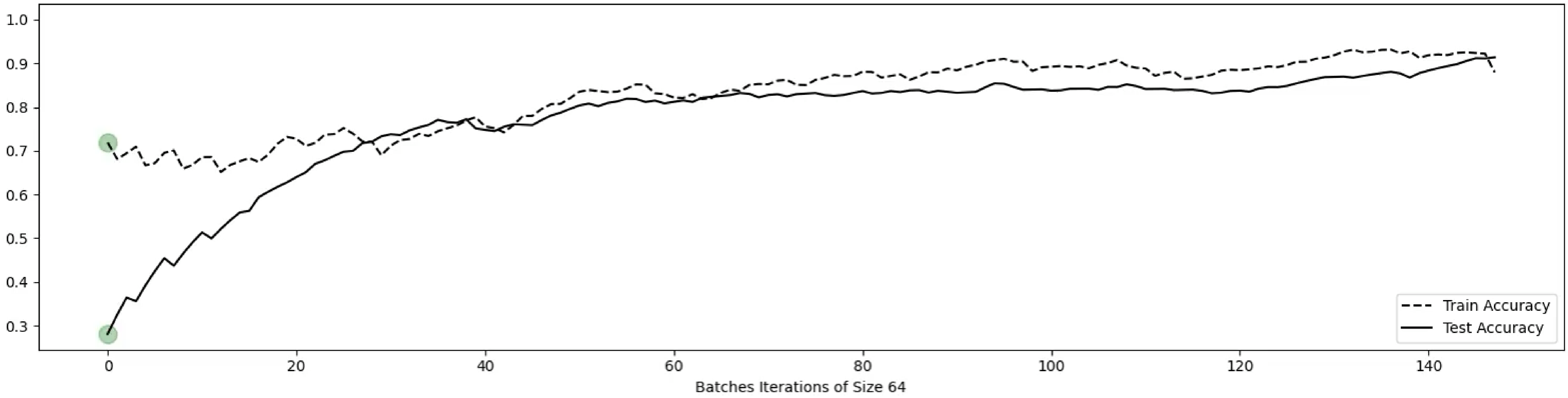
Model Explainability: Integrated Gradients

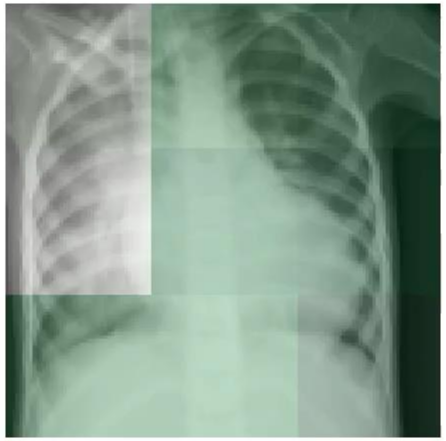


$$(x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial x_i} d\alpha$$

There exits a number of techniques that allow us to highlight the important regions of the input for the models decisions

Model Explainability

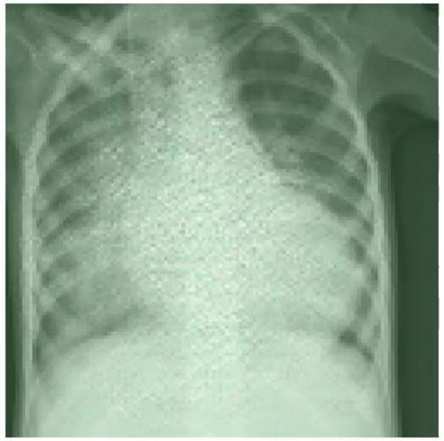




Occlusion

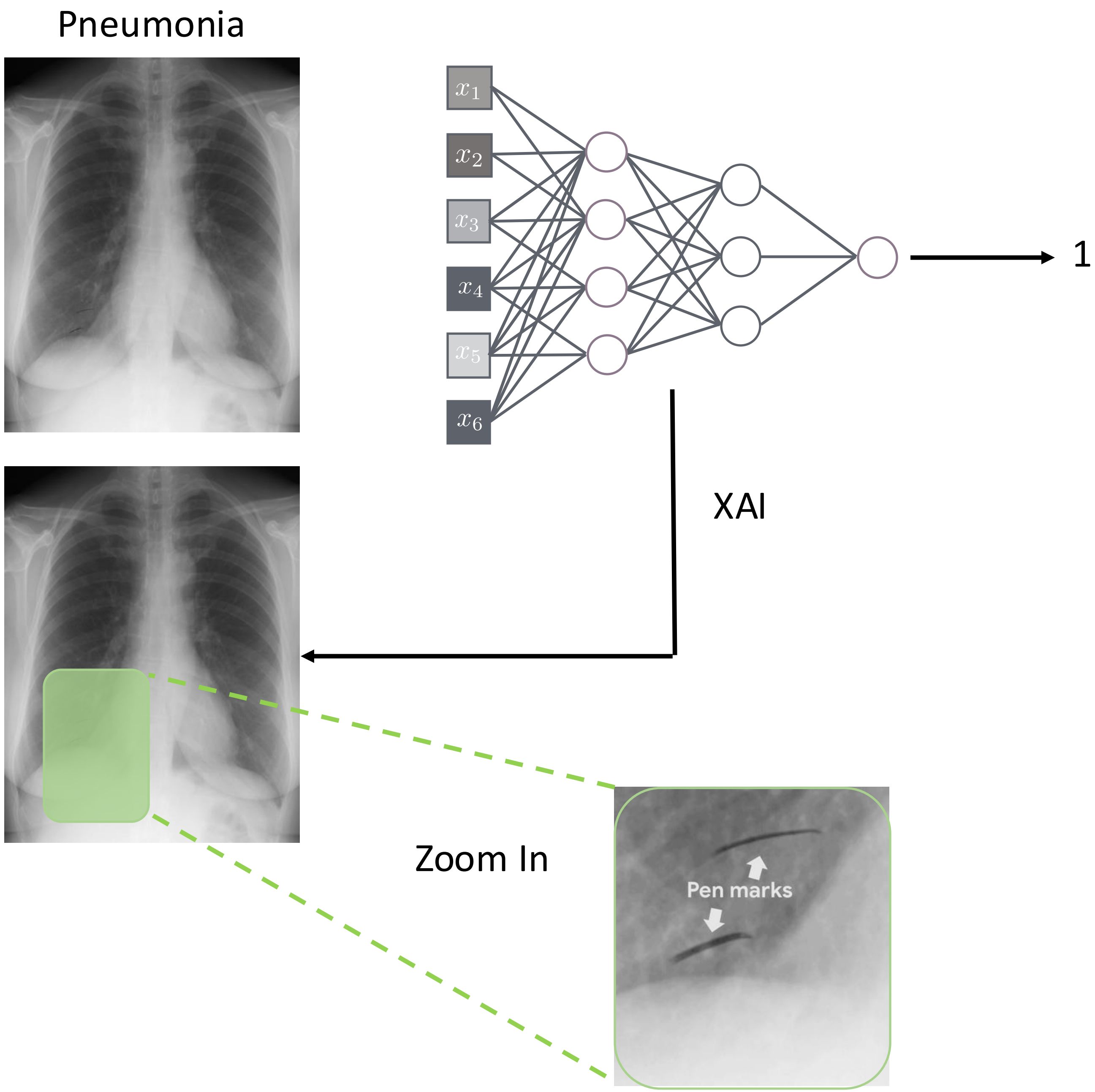
Grad-CAM

Integrated Gradients

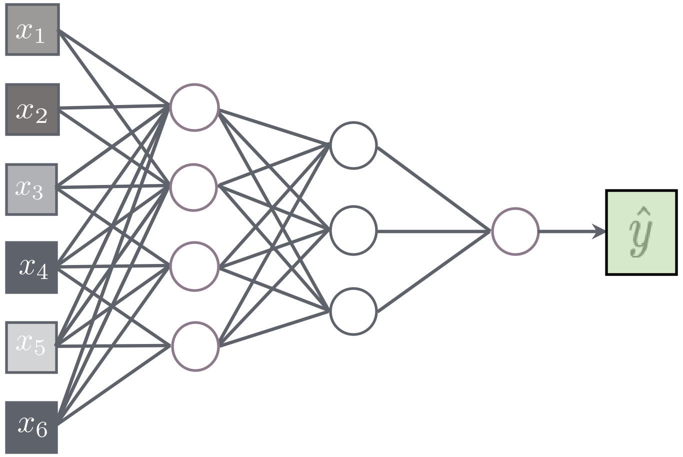


Grad X Input

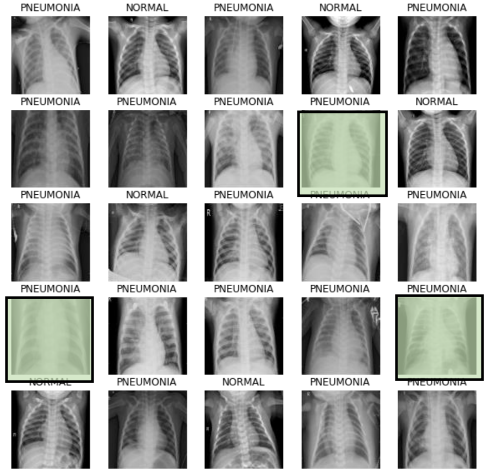
Problem

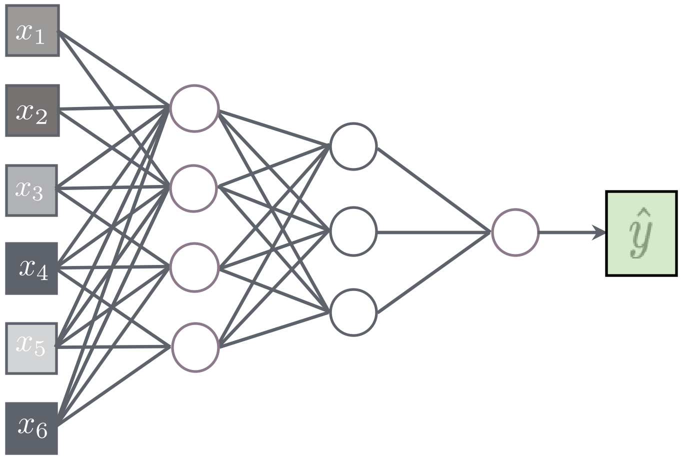


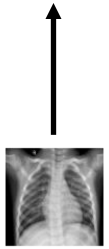
Normal explainability methods
Output is attributed over pixels



Influencer Functions allow us to attribute a model score over full training instances







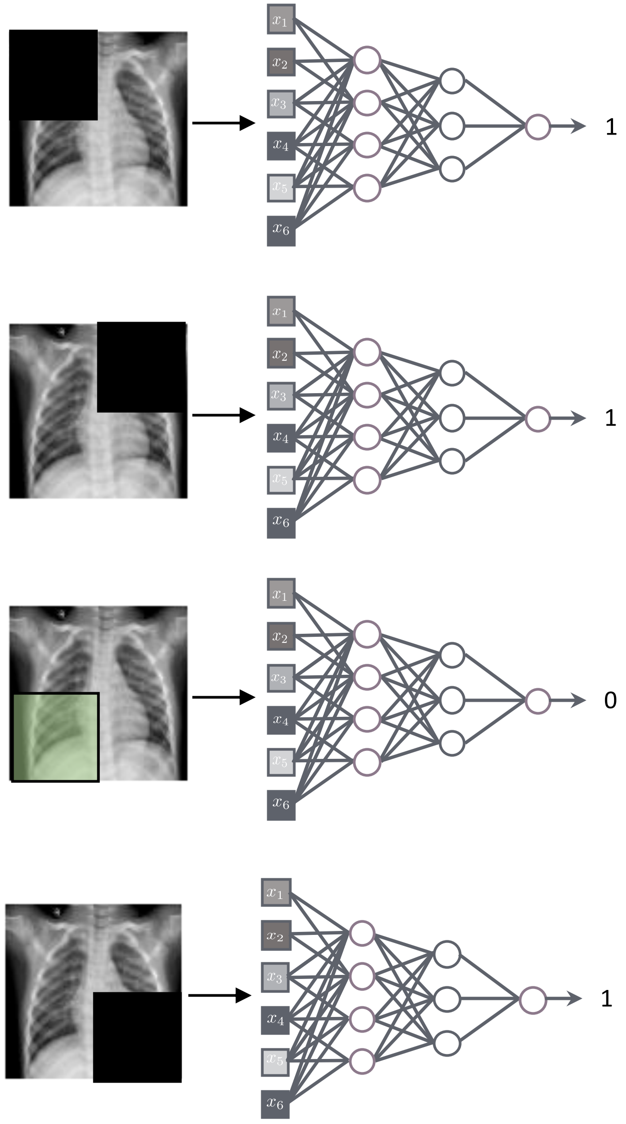
What images during training were useful and adversarial for this instances prediction?

Influence Functions

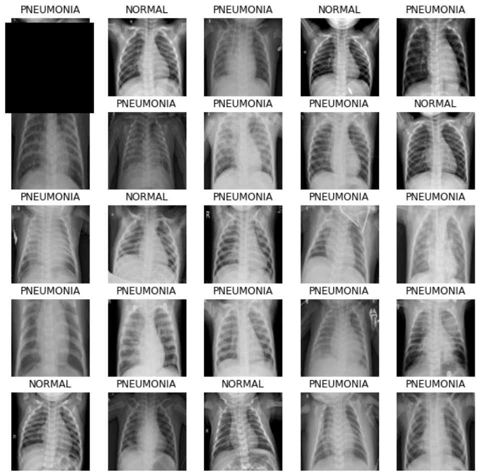
1. Standard XAI allow us to trace a model's predictions back to the pixel values

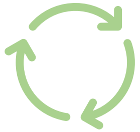
2. Influence functions allow us to trace a model's predictions back to the training data

Occlusion

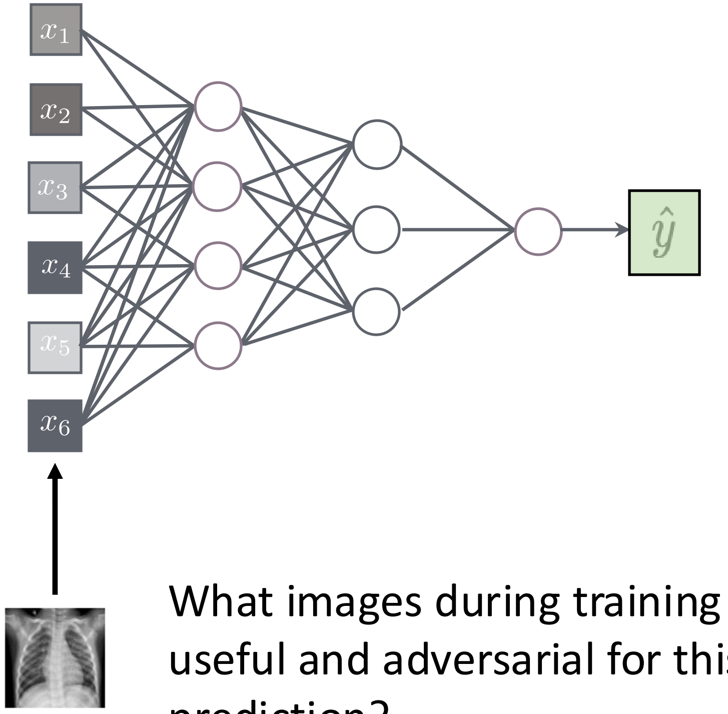


To expensive





Retraining a model n times by removing images one by one is computationally impractical



What images during training were useful and adversarial for this instances prediction?

Influence Functions: Efficient approach

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L\left(z_i, \theta\right)$$

Model is trained to find the optimal parameters that minimize the empirical risk

$$\hat{\theta}_{\epsilon,z} \stackrel{\text{def}}{=} \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L\left(z_i, \theta\right) + \epsilon L(z, \theta)$$

First estimate how much the model parameters have to change if we upweight one of the training samples by a small amount

$$I_{\text{up,params}}\left(z\right) \stackrel{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta})$$

Influence of z point on model weights (classical result)
Would the model have to change drastically to be optimal or not, measure of point importance.

$$I_{\text{up,loss}}\left(z, z_{\text{test}}\right) = \boxed{-\nabla_{\theta} L\left(z_{\text{test}}, \hat{\theta}\right)^{T}} \boxed{H_{\hat{\theta}}^{-1}} \boxed{\nabla_{\theta} L(z, \hat{\theta})}$$

Point z's impact on a test prediction of point z_test

How sensitive is the prediction of our test point to changes in the models parameters

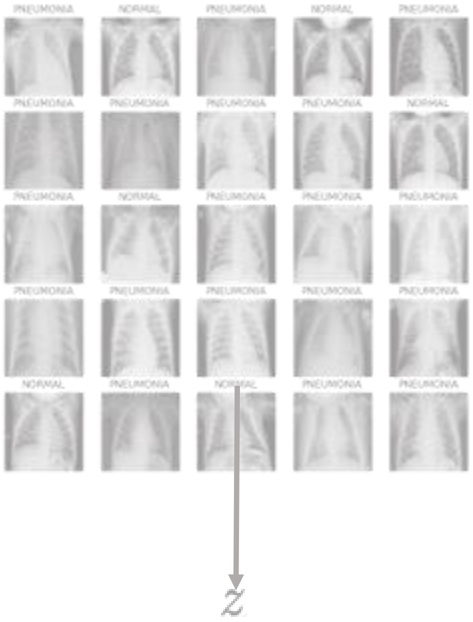
How much the training point influences the models parameters

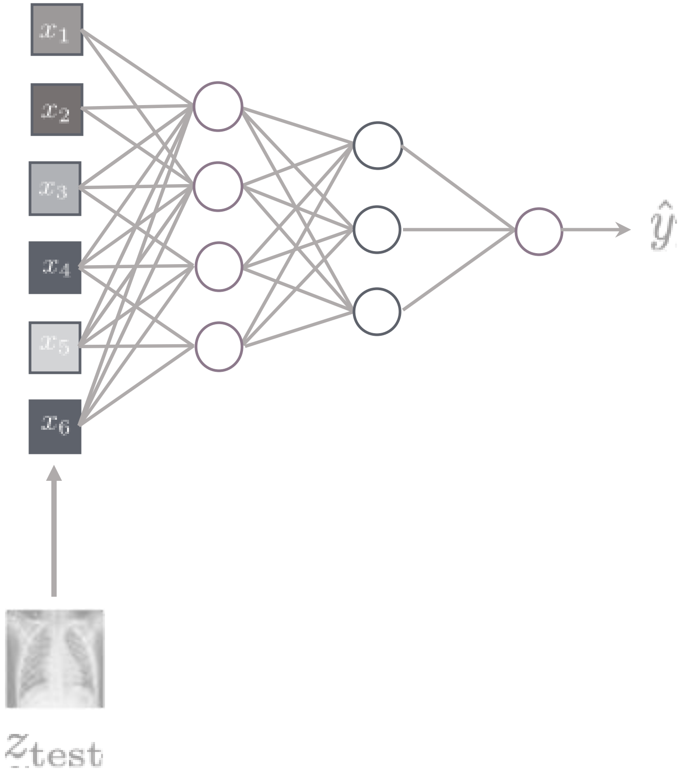
Models stiffness/resistance to change

Point z's impact on a test prediction of point z_test

Notice we never have to retrain the model, all we have to do is take derivatives

Influence Functions: Efficient approach





Instead of removing, upweight z in the loss function

1. How sensitive is the prediction of our test point to changes in the models parameters

2. Model's stiffness/resistance to change

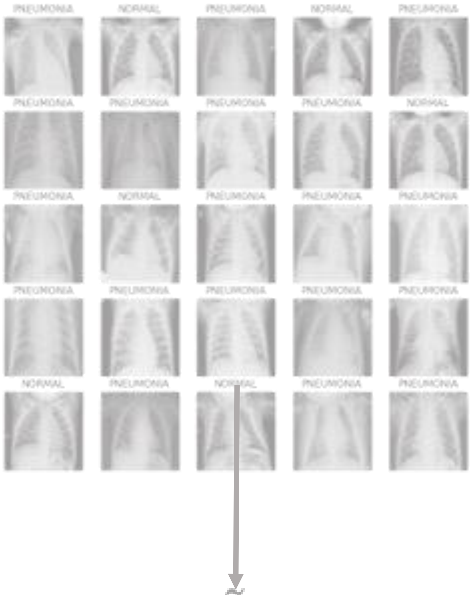
3. How much the training point influences the models parameters

No need to retrain the entire model, we just need to take derivatives!

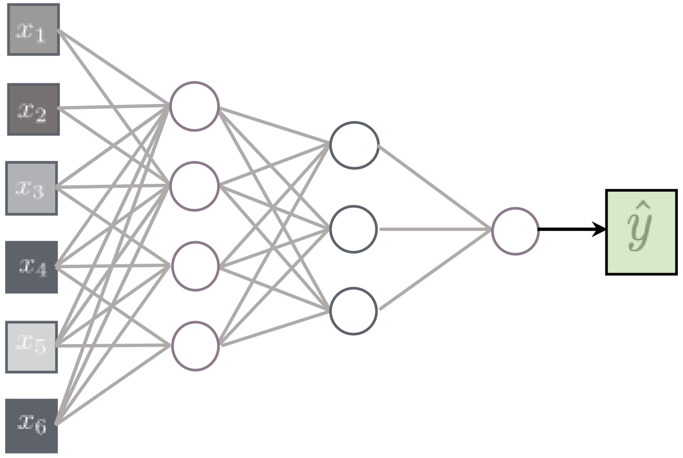
An efficient occlusion method

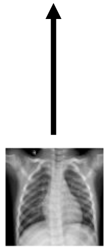
$$I_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

Closed expression for the change in loss wrt a test point when upweighting one of the training examples



z





z_{test}

Instead of removing, upweight z in the loss function

1. How sensitive is the prediction of our test point to changes in the models parameters

2. Model's stiffness/resistance to change

3. How much the training point influences the models parameters

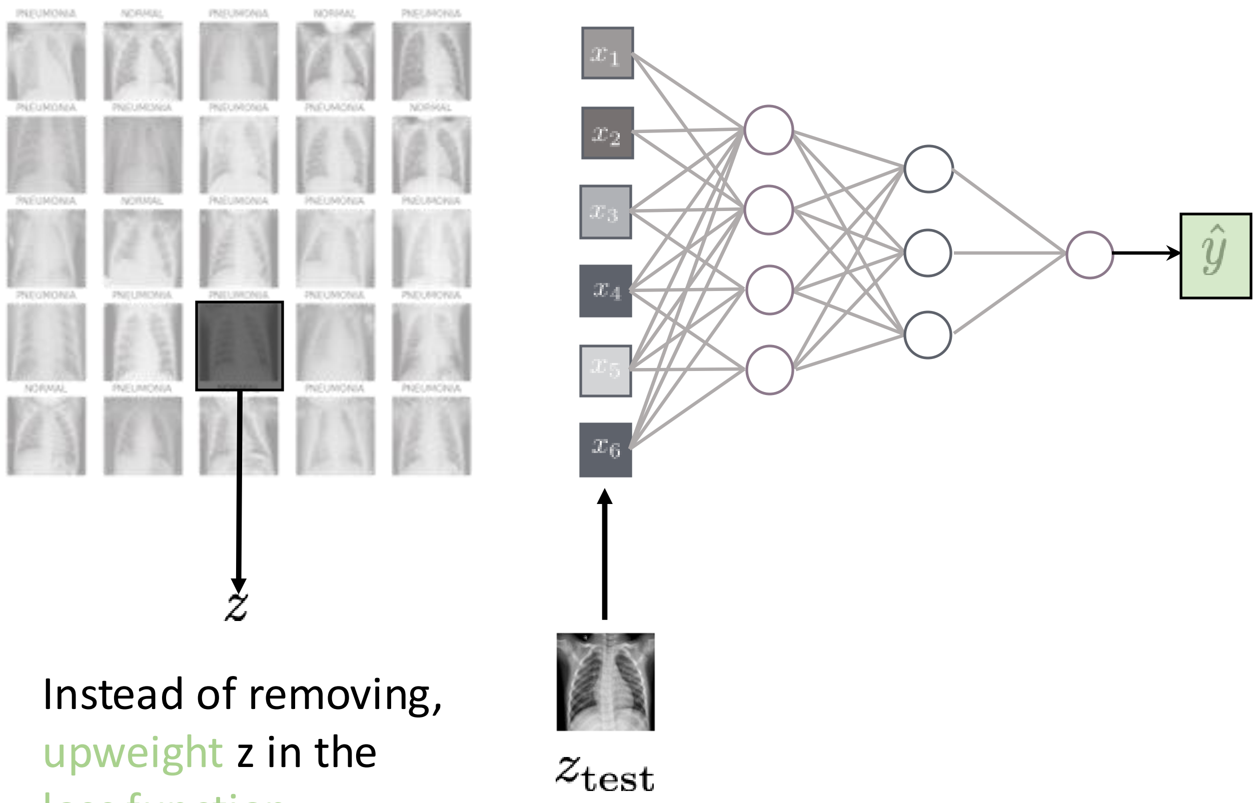
No need to retrain the entire model, we just need to take derivatives!

An efficient occlusion method

$$I_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

Closed expression for the change in loss wrt a test point when upweighting one of the training examples

Influence Functions: Efficient approach



Instead of removing, upweight z in the loss function

1. How sensitive is the prediction of our test point to changes in the models parameters
2. Model's stiffness/resistance to change
3. How much the training point influences the models parameters

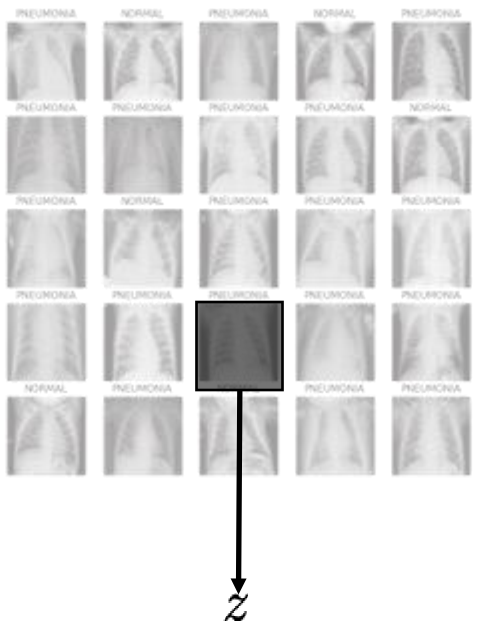
No need to retrain the entire model, we just need to take derivatives!

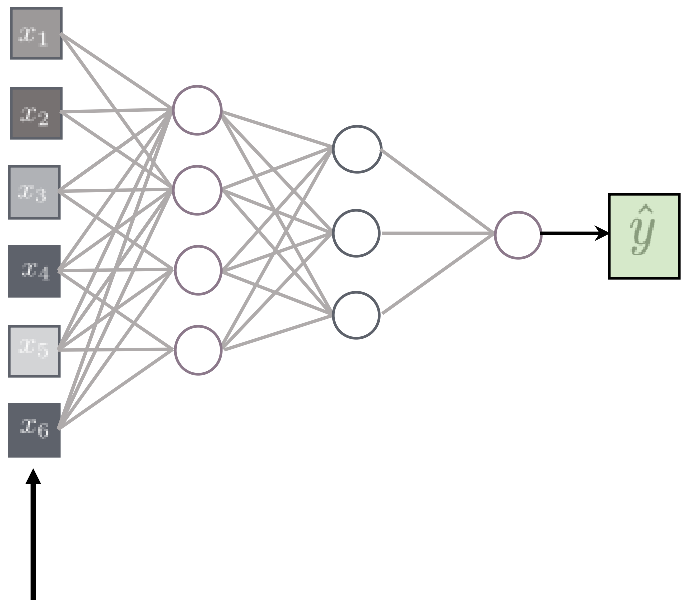
An efficient occlusion method

$$I_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

Closed expression for the change in loss wrt a test point when upweighting one of the training examples

Influence Functions: Efficient approach





1. How sensitive is the prediction of our test point to changes in the models parameters

2. Model's stiffness/resistance to change

3. How much the training point influences the models parameters

Instead of removing, upweight z in the loss function

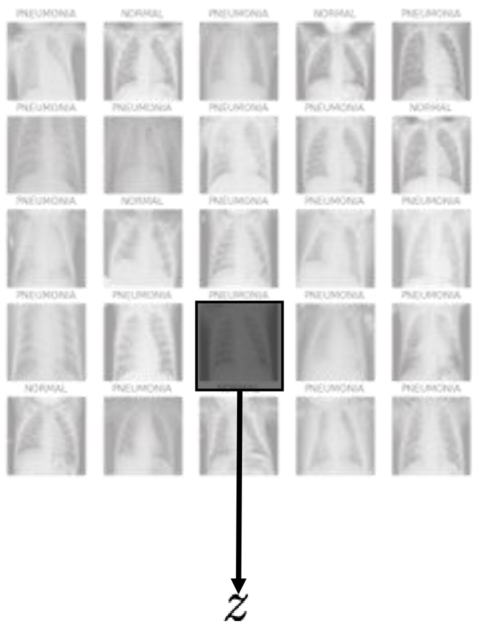
No need to retrain the entire model, we just need to take derivatives!

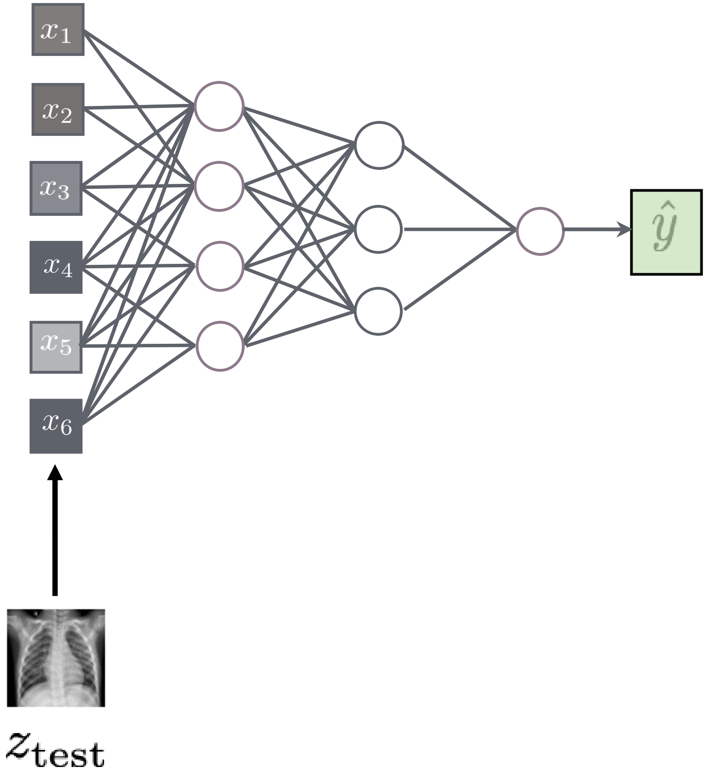
An efficient occlusion method

$$I_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

Closed expression for the change in loss wrt a test point when upweighting one of the training examples

Influence Functions: Efficient approach





Instead of removing, upweight z in the loss function

1. How sensitive is the prediction of our test point to changes in the models parameters
2. Model's stiffness/resistance to change
3. How much the training point influences the models parameters

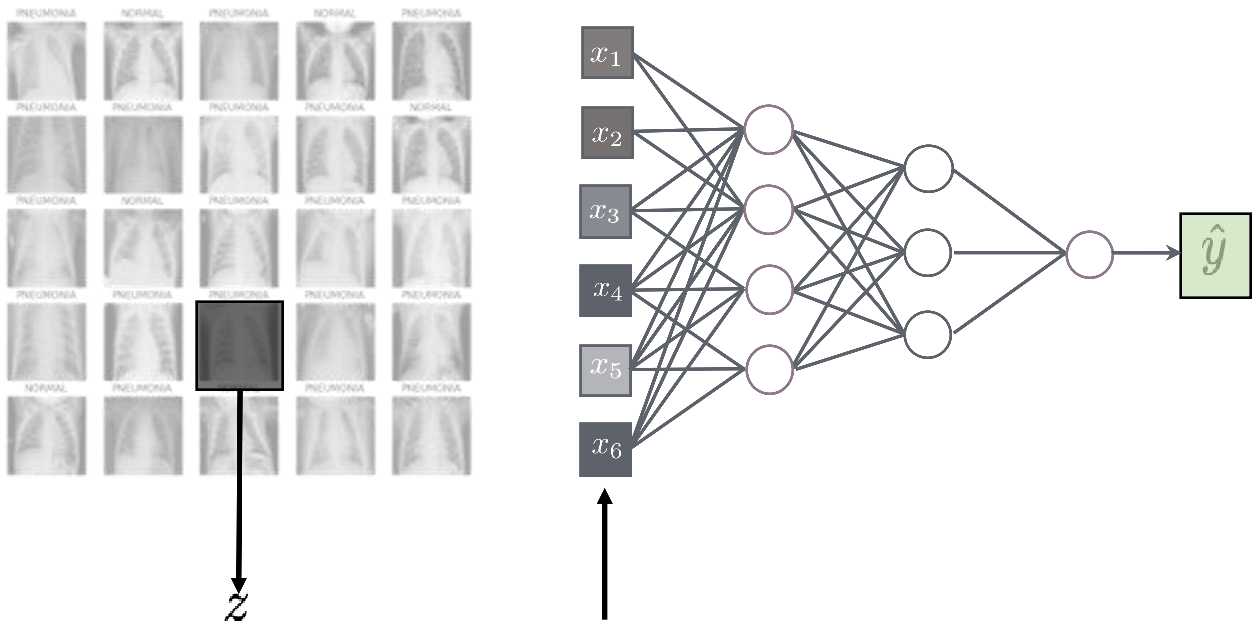
No need to retrain the entire model, we just need to take derivatives!

An efficient occlusion method

$$I_{\text{up,loss}}(z, z_{\text{test}}) = \boxed{-\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^T} H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

Closed expression for the change in loss wrt a test point when upweighting one of the training examples

Influence Functions: Efficient approach



z_{test}

1. How sensitive is the prediction of our test point to changes in the models parameters

2. Model's stiffness/resistance to change

3. How much the training point influences the models parameters

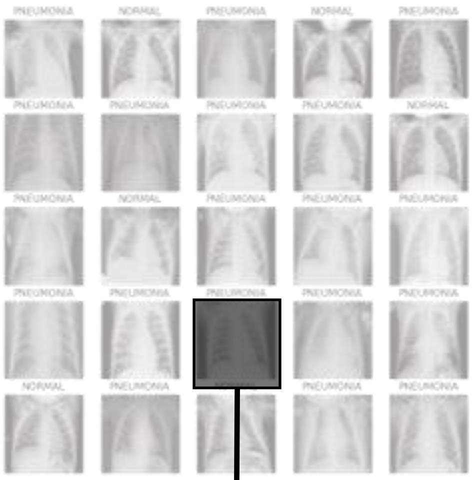
No need to retrain the entire model, we just need to take derivatives!

An efficient occlusion method

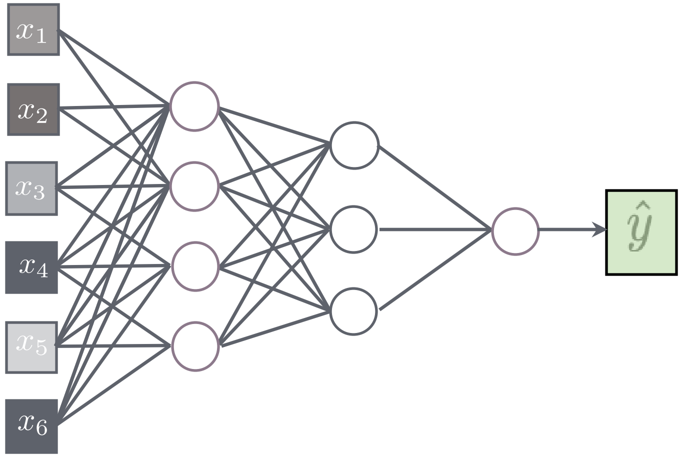
Instead of removing, upweight z in the loss function

$$I_{\text{up,loss}}(z, z_{\text{test}}) = \boxed{-\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^T} \boxed{H_{\hat{\theta}}^{-1}} \nabla_\theta L(z, \hat{\theta})$$

Closed expression for the change in loss wrt a test point when upweighting one of the training examples



z



z_{test}

Instead of removing, upweight z in the loss function

1. How sensitive is the prediction of our test point to changes in the models parameters

2. Model's stiffness/resistance to change

3. How much the training point influences the models parameters

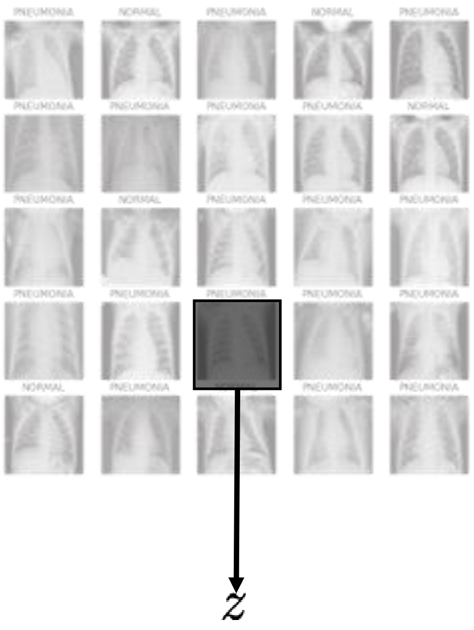
No need to retrain the entire model, we just need to take derivatives!

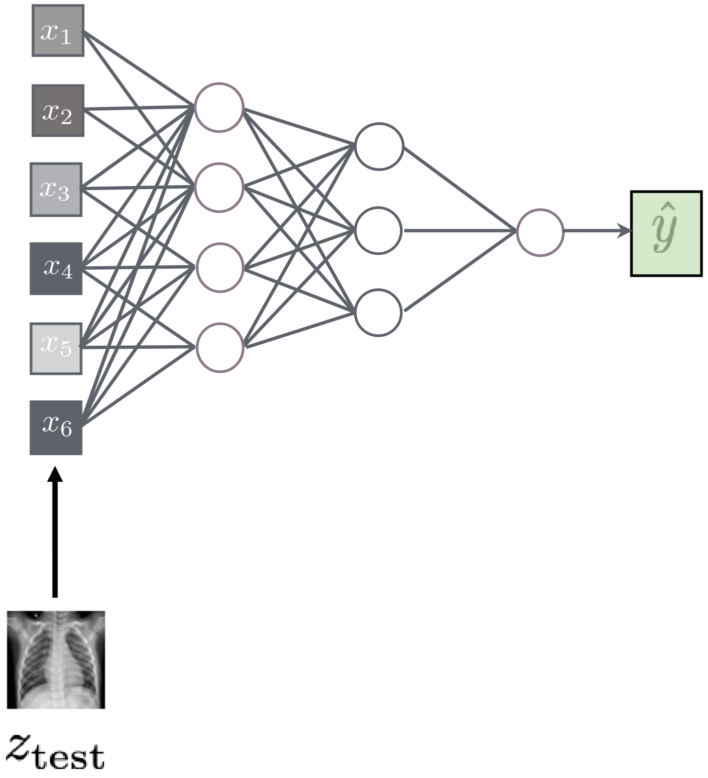
An efficient occlusion method

$$I_{\text{up,loss}}(z, z_{\text{test}}) = \boxed{-\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^T} \boxed{H_{\hat{\theta}}^{-1}} \boxed{\nabla_\theta L(z, \hat{\theta})}$$

Closed expression for the change in loss wrt a test point when upweighting one of the training examples

Influence Functions: Efficient approach





Instead of removing, upweight z in the loss function

1. How sensitive is the prediction of our test point to changes in the models parameters

2. Model’s stiffness/resistance to change

3. How much the training point influences the models parameters

No need to retrain the entire model, we just need to take derivatives!

An efficient occlusion method

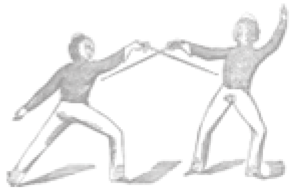
$$I_{\text{up,loss}}\left(z, z_{\text{test}}\right) = \boxed{-\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^T} \boxed{H_{\hat{\theta}}^{-1}} \boxed{\nabla_\theta L(z, \hat{\theta})}$$

Closed expression for the change in loss wrt a test point when upweighting one of the training examples

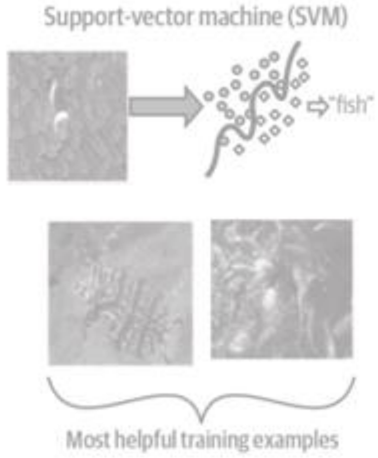
Influence Functions: What is it good for?

- Identify mislabeled data

- Identifying adversarial correctly labeled examples



- Understanding model behaviour



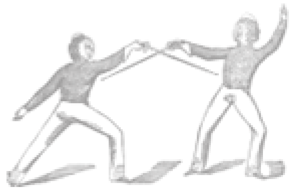
Lower-level features such as contrast and texture seem to have higher influence for SVM

higher-level features such as shape and patterns seem to be important for InceptionN

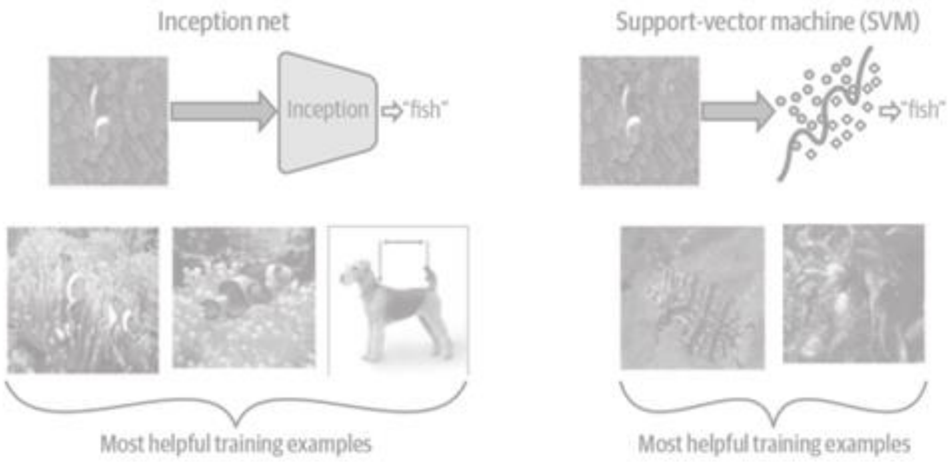
Influence Functions: What is it good for?

1 Identify mislabeled data

Identifying adversarial correctly labeled examples



Understanding model behaviour



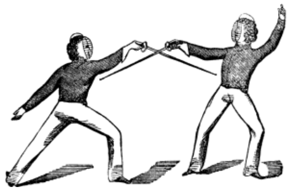
Lower-level features such as contrast and texture seem to have higher influence for SVM

higher-level features such as shape and patterns seem to be important for InceptionN

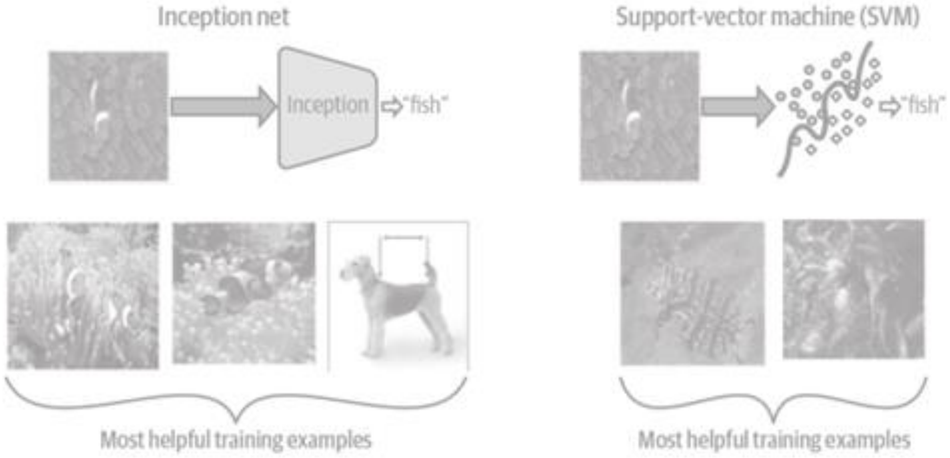
Influence Functions: What is it good for?

1 Identify mislabeled data

2 Identifying adversarial correctly labeled examples



Understanding model behaviour



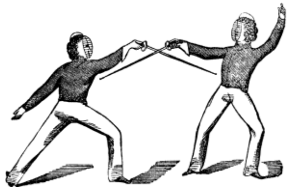
Lower-level features such as contrast and texture seem to have higher influence for SVM

higher-level features such as shape and patterns seem to be important for InceptionN

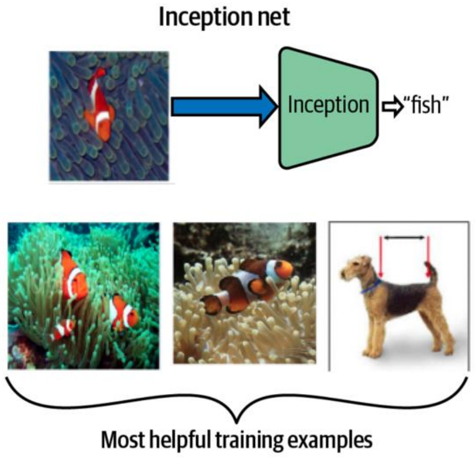
Influence Functions: What is it good for?

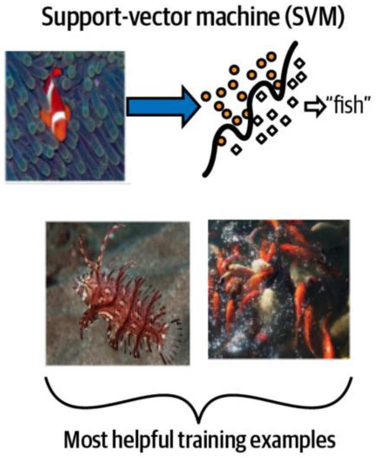
1 Identify mislabeled data

2 Identifying adversarial correctly labeled examples



3 Understanding model behaviour





Lower-level features such as contrast and texture seem to have higher influence for SVM

higher-level features such as shape and patterns seem to be important for InceptionN

Sources

GitHub

https://github.com/kohpangwei/influence-release

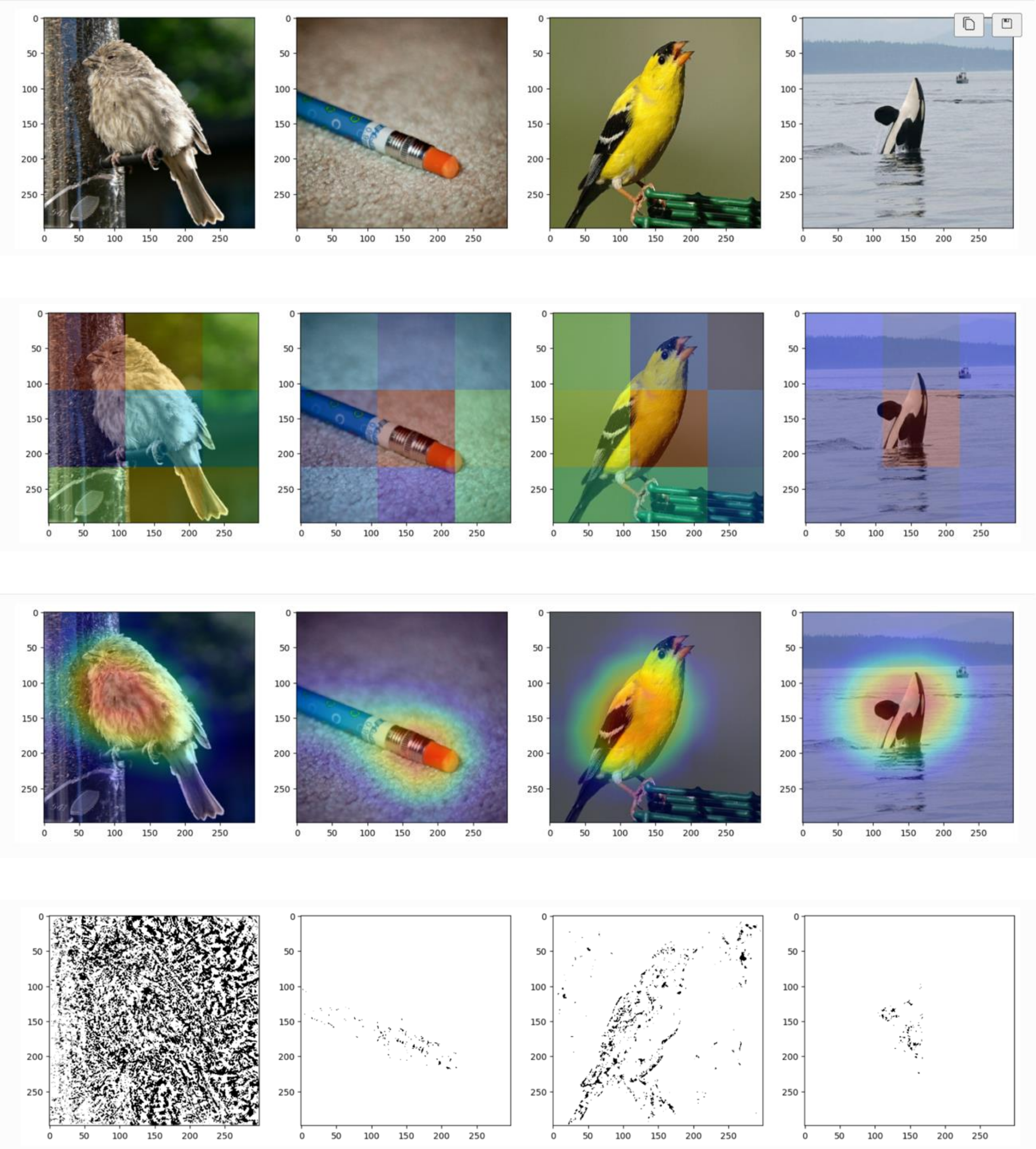
Original paper

https://arxiv.org/pdf/1703.04730

Captum

https://captum.ai/

Example form converge multiclassification dataset



Original Images

Occlusion

Grad-CAM

Integrated Gradients