

Causality and Generative Models

This week's paper is [1]

Author: Anton Zhitomirsky

Contents

1 Trustworthy AI/ML	3
1.1 The need for data	3
1.2 What are the hurdles to getting more data?	4
1.3 Secure and Privacy-aware ML	4
1.4 Secure and Privacy-preserving ML	5
1.5 Federated learning	6
1.5.1 Federated SGD	7
1.5.2 Federated Averaging	8
1.5.3 Differences	8
1.5.4 Algorithm	9
1.5.5 Challenges	9
1.6 Homomorphic Encryption	10
1.6.1 ML in client/server setting	10
1.6.2 Homomorphic Encryption	10
1.6.3 Advantages and Disadvantages in ML	11
1.7 Secure Multi-party Computation	11
1.8 What is Privacy?	12
1.8.1 k-anonymity	12
1.9 Privacy attacks	14
1.9.1 Model inversion	14
1.9.2 Membership inference	15
1.9.3 Attribute inference	15
1.10 Differential Privacy	15
1.10.1 Randomized responses	16
1.10.2 Practical Application of Differential Privacy	17
1.10.3 Concrete Mitigation: stochastic gradient descent (DP-SGD)	18
2 Interpretability and Explainability	19
2.1 Interpretability	19

2.1.1	Why is it important?	19
2.1.2	Common misunderstandings	21
2.2	How can we interpret an existing ML model?	22
2.2.1	Occlusions	24
2.2.2	Saliency maps	25
2.2.3	Salicency maps - Gradient (backpropagation)	27
2.2.4	Salicency maps - Guided backpropagation	27
2.3	Cam and Grad-Cam	29
2.4	DeepDream / Inceptionism	30
2.5	Inversion	31
3	Robustness: Adversarial Methods	32
3.1	Adversarial Attacks	32
3.1.1	Perturbation	33
3.1.2	Fast Gradient Sign Method	35
3.2	Adversarial attacks	36
A	References	36
Bibliography		37

1 Trustworthy AI/ML

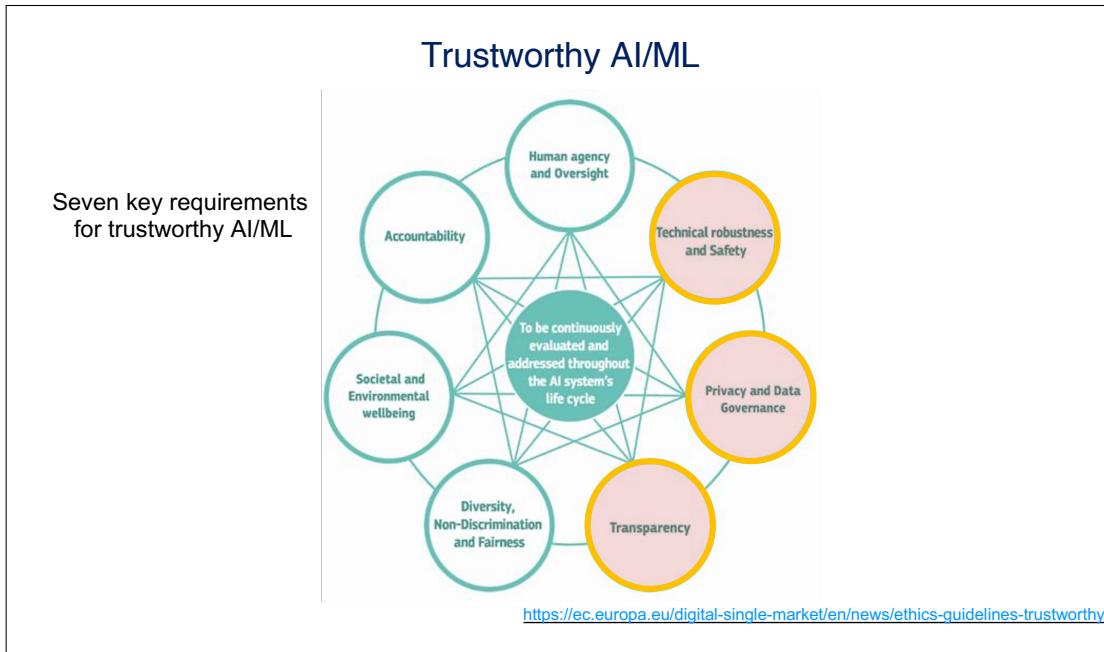


Figure 1: Some of these concepts don't relate to technical aspects, but general. This is the definition of the EU of what makes AI trustworthy and Reliable.

1.1 The need for data

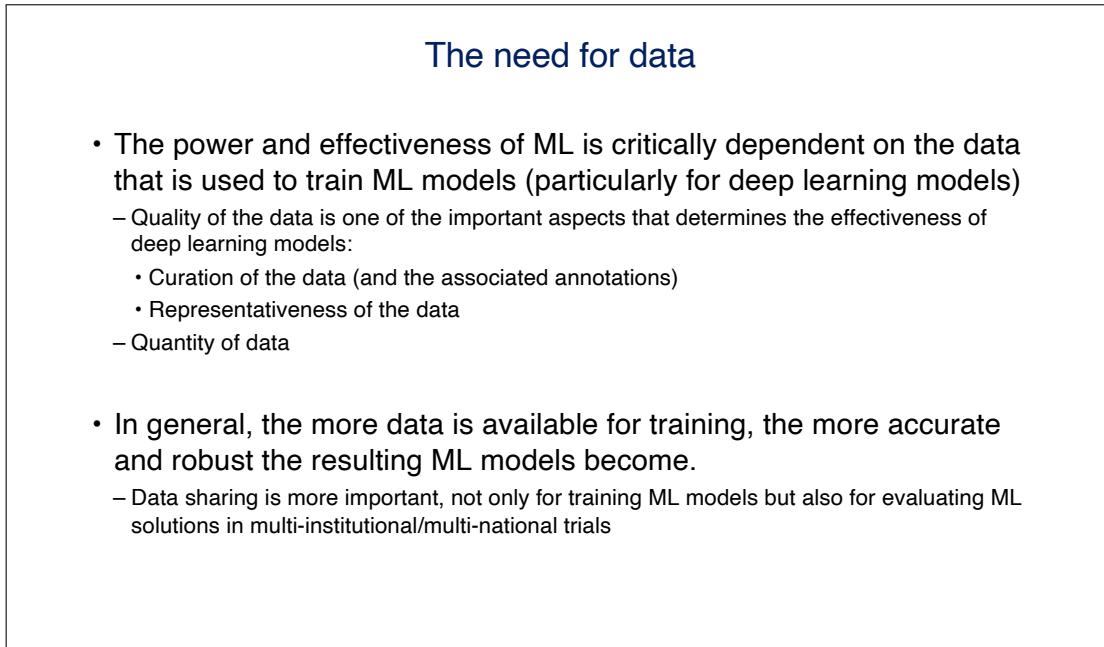


Figure 2: In order to build systems which work well we would like to use as much data as possible to train these models. However, the quality of the data also needs to be good (requires diversity, representative of problem and applies to the annotations).

1.2 What are the hurdles to getting more data?

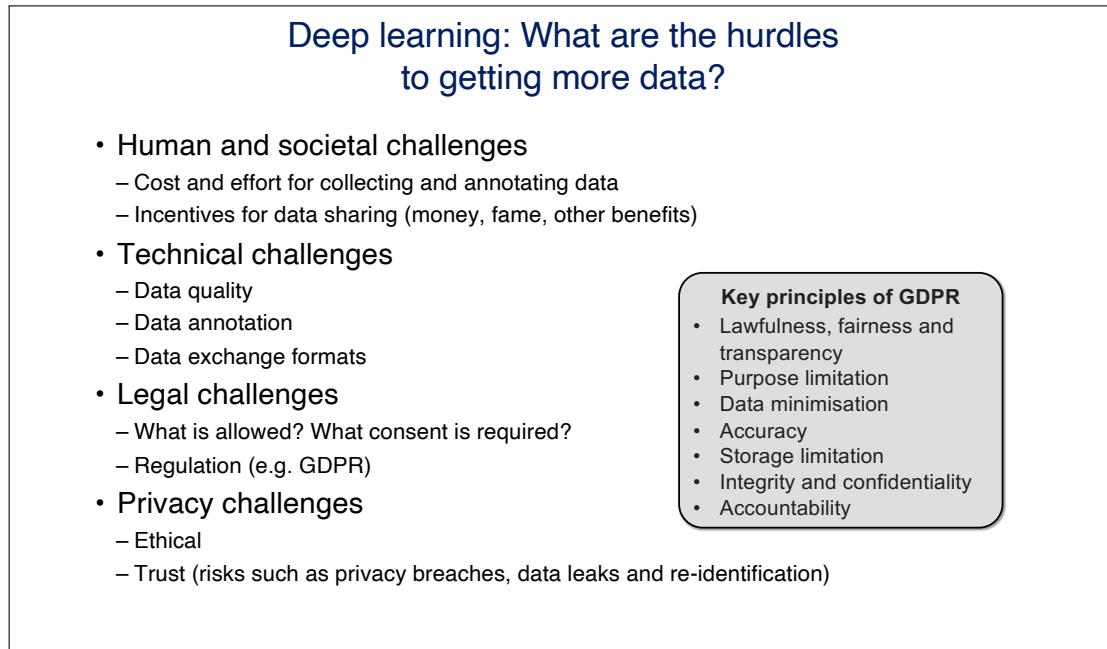


Figure 3: For different quality of data and annotation, we can weight these differently in the loss function.

1.3 Secure and Privacy-aware ML

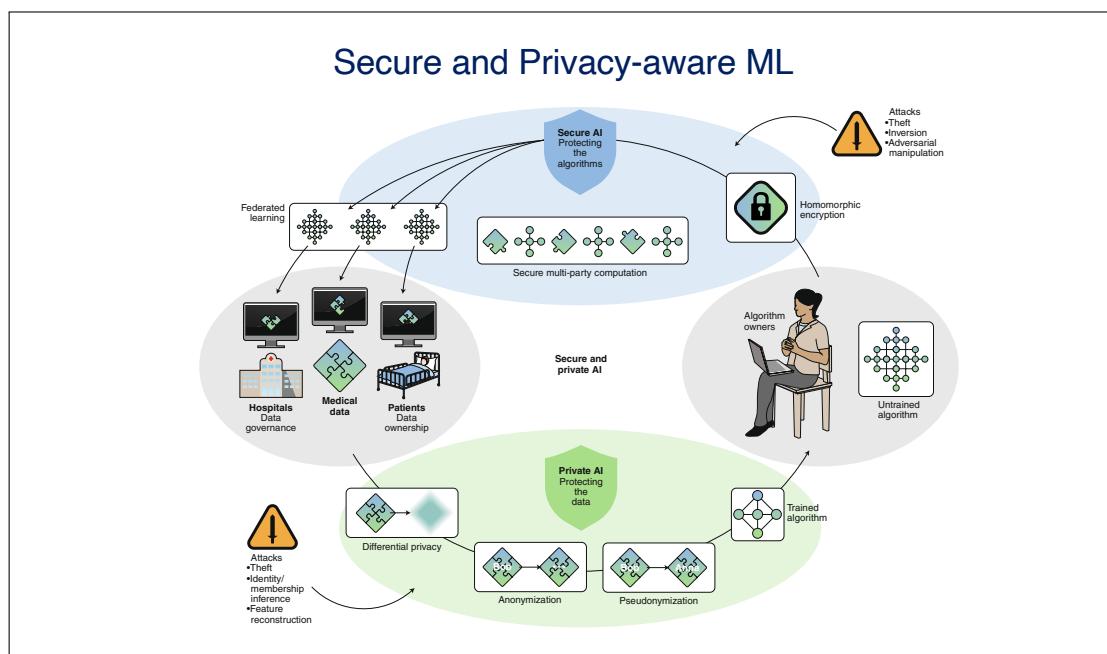


Figure 4: The concepts here apply to all imaging problems. Federated learning controls the flow of data. All of these techniques attempt to protect the computation.

1.4 Secure and Privacy-preserving ML

Secure and privacy-preserving ML

- Optimal privacy preservation requires implementations that are secure by default so-called *privacy by design*

- Requirements:

- Minimal or no data transfer
- Provision of theoretical and/or technical guarantees of privacy

Federated learning: train a ML model across decentralized clients with local data, without exchanging them

Differential privacy: perturb the data so that information about the single individual is reduced while retaining the capability of learning

Figure 5: We can't retrospectively take an approach and hope you can build techniques which make sure that you ensure privacy. You need to design it from first principles and there are two overriding principles for this privacy by design. These are above. For minimal data transfer you can use federated learning, this uses decentralized learning with a bunch of clients with their data locally. The data is never transferred to the server, but only the result. Here, federated learning doesn't really guarantee privacy, but it gives the user control of their data. The second part, is differential privacy which anonymised the data.

Secure and privacy-preserving ML

- Optimal privacy preservation requires implementations that are secure by default so-called *privacy by design*

- Requirements:

- Minimal or no data transfer
- Provision of theoretical and/or technical guarantees of privacy

- Other approaches for privacy-preserving AI:

- Homomorphic encryption which enables learning from encrypted data
- Secure multi-party computing where processing is performed on encrypted data shares, split among them in a way that no single party can retrieve the entire data on their own.
- Trusted execution environments

Figure 6: There are other add-ons you can use. The techniques are listed above.

1.5 Federated learning

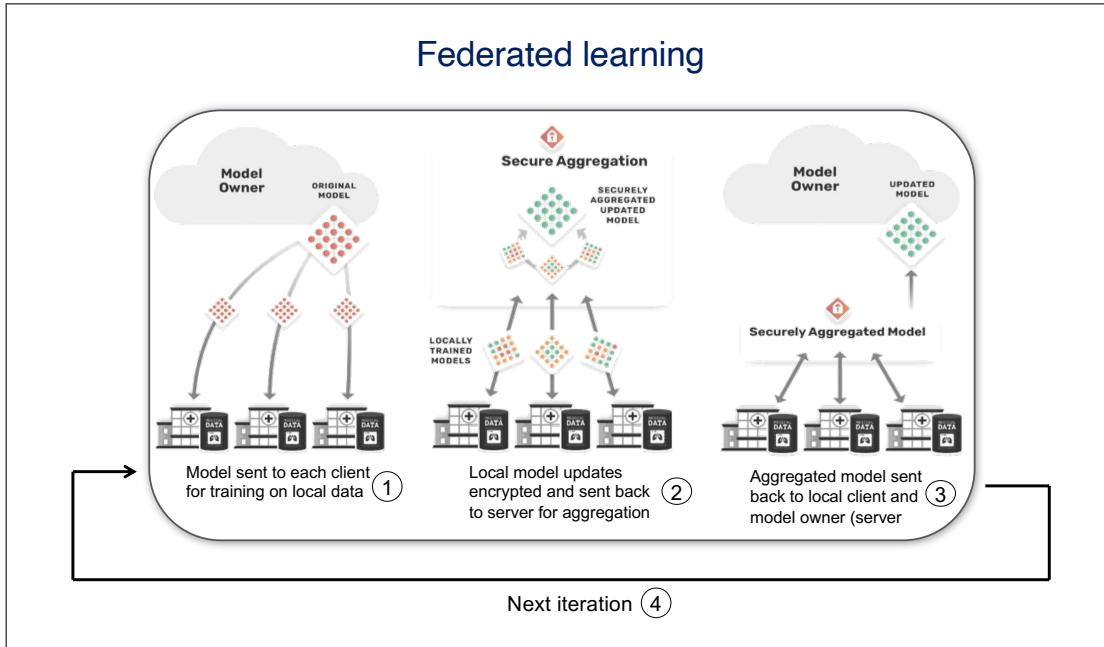


Figure 7: I have a central server where each client sends their data to a central cite. The data is sensitive, and the data owners are not allowed by law to share their data. Federated learning makes you send your ML model to each of the cite and each cite update the model parameters and sends it back to the central cite. The central cite does some aggregation and you repeat the process. Importantly, is that the data stays with the owners.

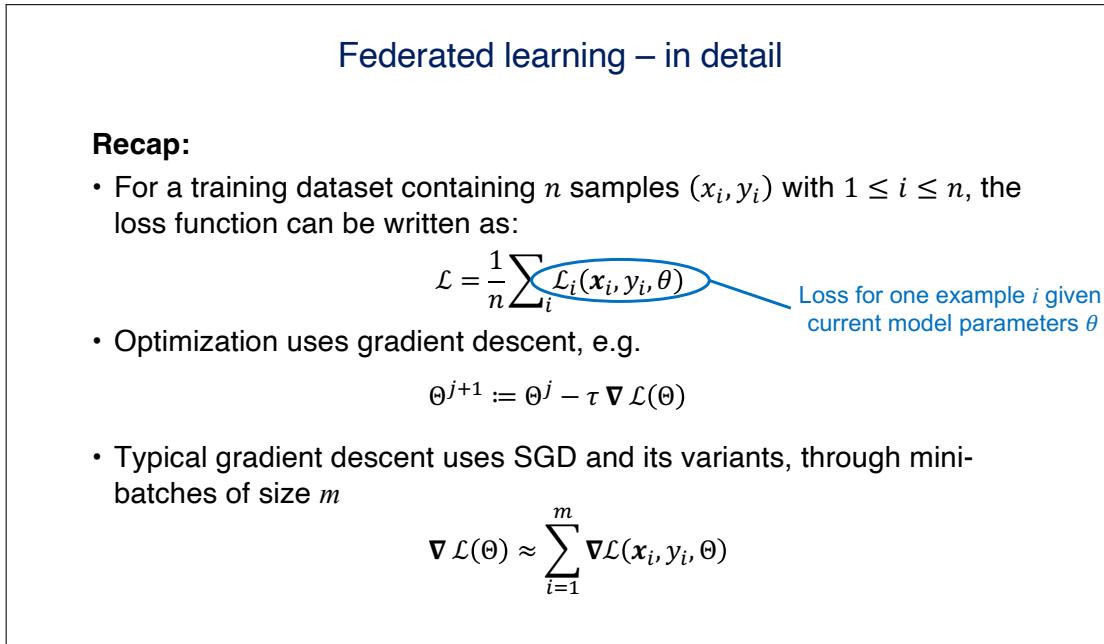


Figure 8: This is the standard learning process. If using stochastic gradient descent you approximate your gradient by just looping over all the samples in the minibatch.

Federated learning – in detail

In federated learning:

- Suppose N training samples are distributed to K clients, P_k is the set of indices of samples at client k , and $n_k = |P_k|$

$$\mathcal{L}(\Theta) = \sum_{k=1}^K \frac{n_k}{N} \mathcal{L}_k(\Theta)$$

with

$$\mathcal{L}_k(\Theta) = \frac{1}{n_k} \sum_{i \in P_k} \mathcal{L}(\mathbf{x}_i, y_i, \Theta)$$

$$\mathbb{E}_{P_k}[\mathcal{L}_k] = \mathcal{L} \quad \text{iid setting}$$

$$\mathbb{E}_{P_k}[\mathcal{L}_k] \neq \mathcal{L} \quad \text{non-iid setting}$$

Figure 9: If you do federated learning, you assume that the training samples are distributed to K clients. You can still compute the loss function by computing the loss function from each client by weighting it depending on how many or how big the training set is. The first equation is the standard loss PER CLIENT. If you sum this, you get the standard loss function.

If you compute the loss this way, the expectation of this loss computed this way for each client is equal to the overall loss if you assume the iid setting. If you have huge amounts of data in homogeneity lets say (a hospital in US may have different population characteristics to hospitals in EU), then each individual loss terms may not be in the expectation equivalent to the overall loss. This impacts how you train.

1.5.1 Federated SGD

Federated SGD

In federated learning:

- Suppose a C fraction of clients are selected at each round:
 - $C = 1$: full-batch (non-stochastic) gradient descent
 - $C < 1$: stochastic gradient descent (SGD)

- Each client computes:

$$\nabla \mathcal{L}_k(\Theta)$$

- Server computes:

$$\nabla \mathcal{L}(\Theta) = \sum_{k=1}^K \frac{n_k}{N} \nabla \mathcal{L}_k(\Theta)$$

Figure 10: Assume that each client independently computes the loss and derivative for their dataset. What you transmit is the gradient

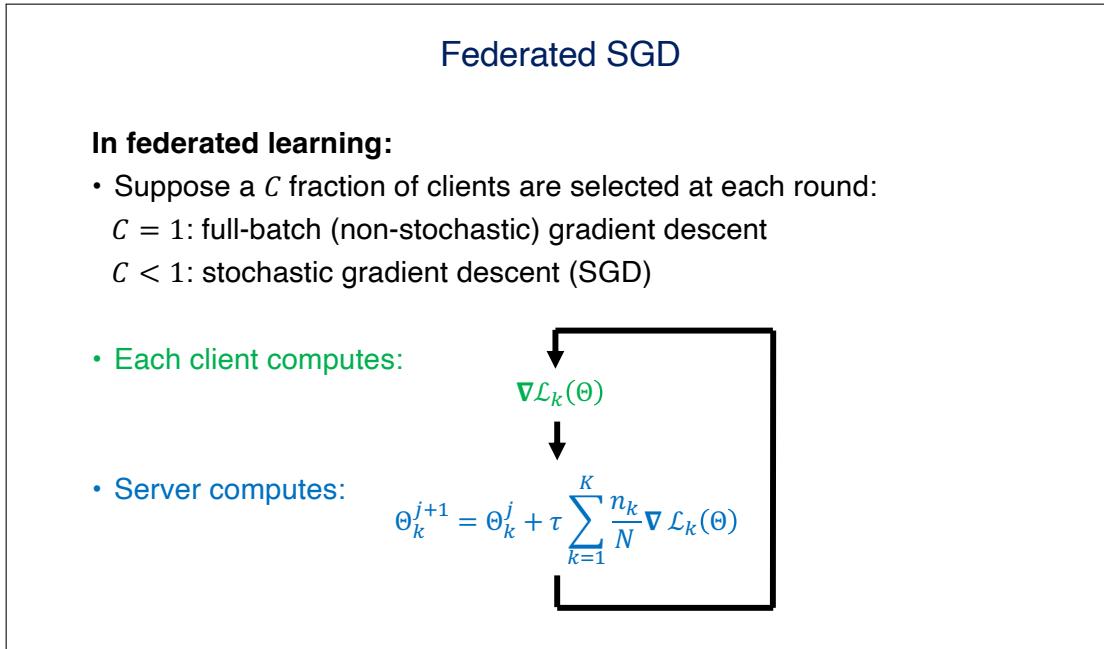


Figure 11: Then the server updates the model after combining all gradients.

1.5.2 Federated Averaging

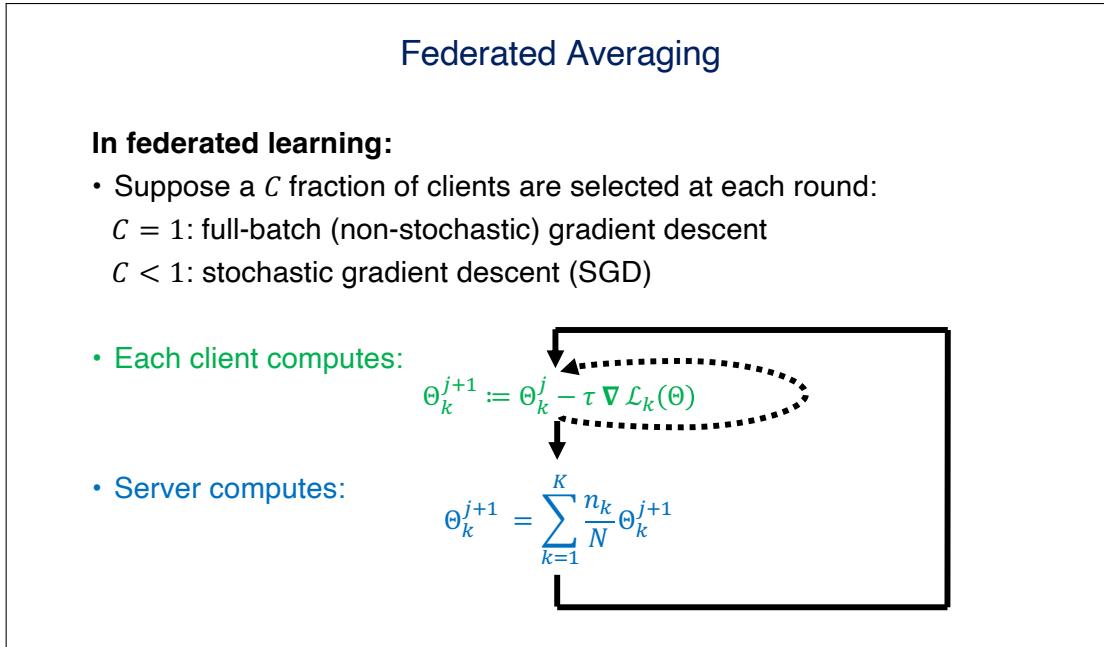


Figure 12: There is an alternative approach; the client transfers the gradient to the server, but you can also locally update the parameters by performing on each client a gradient descent separately. Then generating a new set of parameters and iterating a few times. Then sending out to the server back an updated set of parameters. On the server side, you then update the parameters by computing a weighted average of the parameters

1.5.3 Differences

The difference between the two approaches is that each client may not have that much compute power, and each client may typically have only one dataset. It doesn't make sense to do gradient descent on my local set because I have a very noisy gradient. This is because I'm limited to a batch size of one. In

the first approach, you send the gradient, but you accumulate the gradient and perform much more reliable gradient descent. However there, you require constant communication between the client and the server. In the second example, you force each client to do 10 iterations, and only then they share updated parameters. There, you have better control over the communication between client and server.

1.5.4 Algorithm

Federated Averaging

Algorithm 1 FederatedAveraging. The K clients are indexed by k ; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

```

initialize  $w_0$ 
for each round  $t = 1, 2, \dots$  do
     $m \leftarrow \max(C \cdot K, 1)$ 
     $S_t \leftarrow (\text{random set of } m \text{ clients})$ 
    for each client  $k \in S_t$  in parallel do
         $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$ 
     $w_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} w_{t+1}^k$ 

ClientUpdate( $k, w$ ): // Run on client  $k$ 
     $\mathcal{B} \leftarrow (\text{split } \mathcal{P}_k \text{ into batches of size } B)$ 
    for each local epoch  $i$  from 1 to  $E$  do
        for batch  $b \in \mathcal{B}$  do
             $w \leftarrow w - \eta \nabla \ell(w; b)$ 
    return  $w$  to server

```

- First, model is randomly initialized on the central server
- For each round t :
 - A random set of clients are chosen;
 - Each client performs local gradient descent steps;
 - The server aggregates model parameters submitted by the clients.

Figure 13: caption

1.5.5 Challenges

Challenges for federated learning

- Non-IID data
 - Training data for a given client is typically site specific, hence the site's local dataset will not be representative of the distribution of training samples.
- Unbalanced data
 - Sites may have a lot or little training data, leading to varying amounts of local training data across different sites.
- Massively distributed data
 - There may be extreme scenarios where each site only has very training samples (in the limiting case one example)
- Communication costs
 - Communication between clients and servers occurs communication overheads. The amount of overhead will depend on the number of clients and the frequency of updates from/to server.

Figure 14: 1) approximation of whole gradients no-longer approximates the overall gradient. 2) updates might not be as reliable on some sites as others 3) this is extreme and inefficient.

1.6 Homomorphic Encryption

There is still an extent of sensitive data being transferred.

- Based on the assumption that one can perform (basic) arithmetic on encrypted values, i.e.

$$[x] \oplus [y] = [x + y] \quad \text{and} \quad [x] \otimes [y] = [x \cdot y]$$

- Here:

$[xyz]$	encryption of some plaintext xyz
\oplus	homomorphic addition operation in ciphertext space
\otimes	homomorphic multiplication operation in ciphertext space

Figure 15: finds an encryption scheme, such that the encrypted data can be manipulated in the encrypted domain as though it had been decrypted.

1.6.1 ML in client/server setting

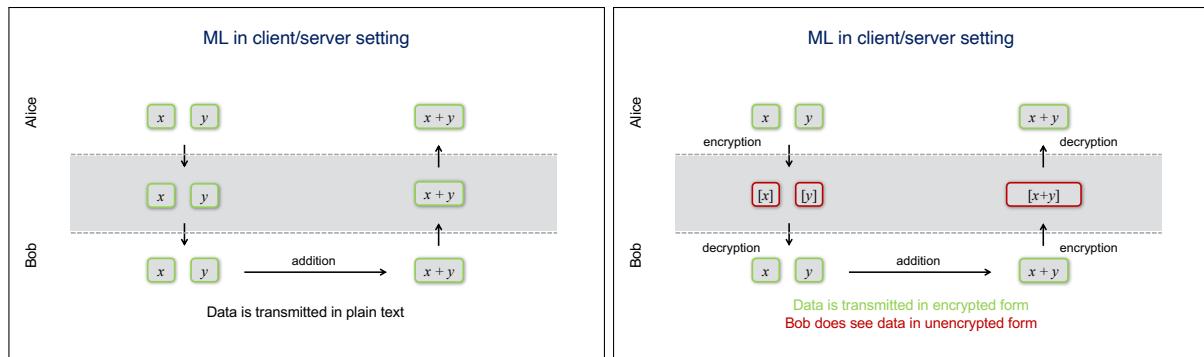


Figure 16: If Alice is the client and Bob is the server, then sending the data to the server, performing the addition, and sending it back is insecure. With standard encryption, we protect it through encryption, then we decrypt it because we need to decrypt it.

1.6.2 Homomorphic Encryption

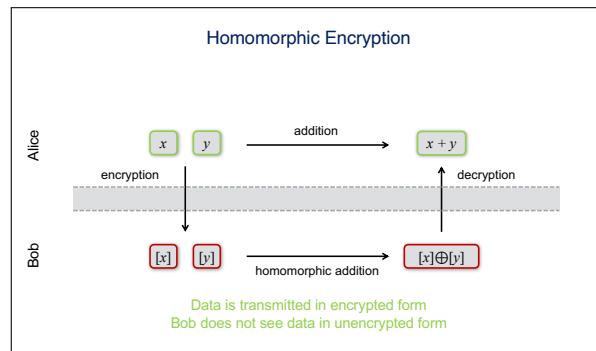
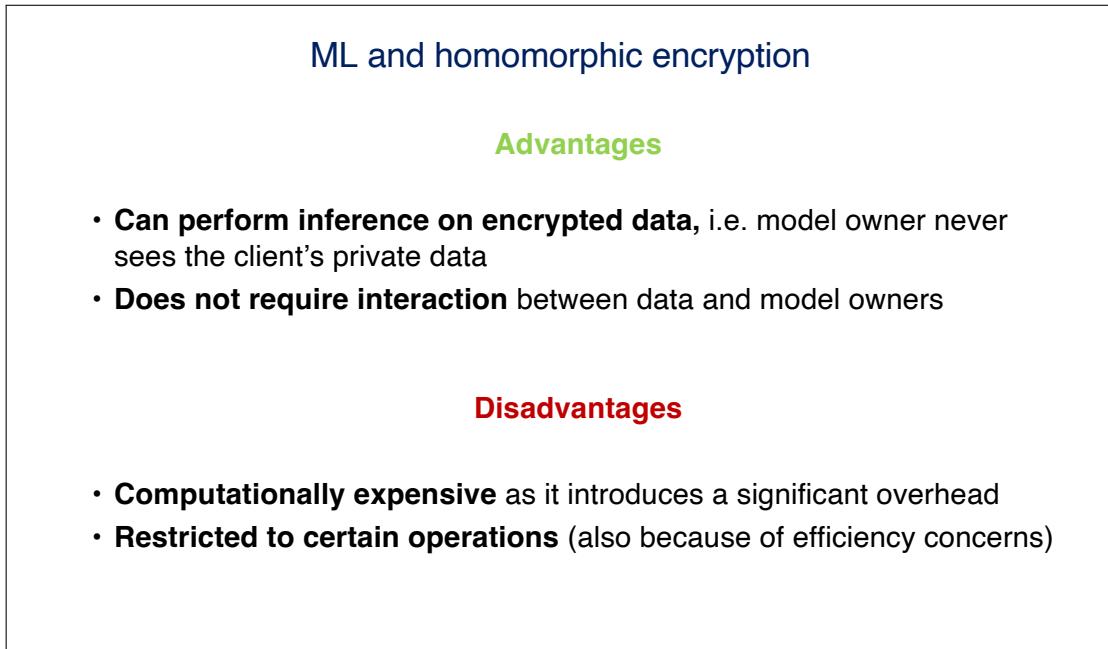
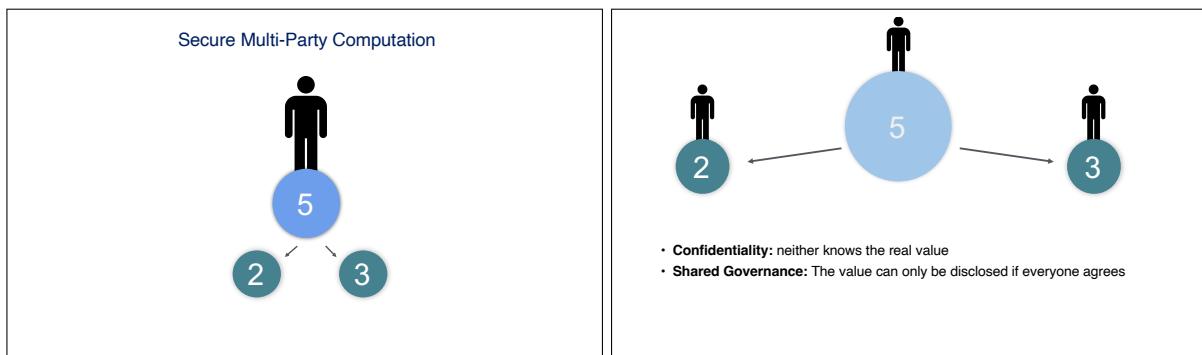


Figure 17: In homomorphic encryption, we can perform the addition in the encrypted domain. Bob (the server) never gets to decrypt the data. However, this is computationally expensive (100x slower for deep learning and also some mathematical operations are hard to find a homomorphic encryption for).

1.6.3 Advantages and Disadvantages in ML

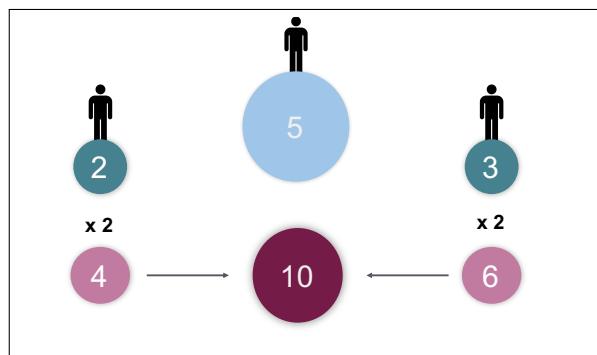


1.7 Secure Multi-party Computation



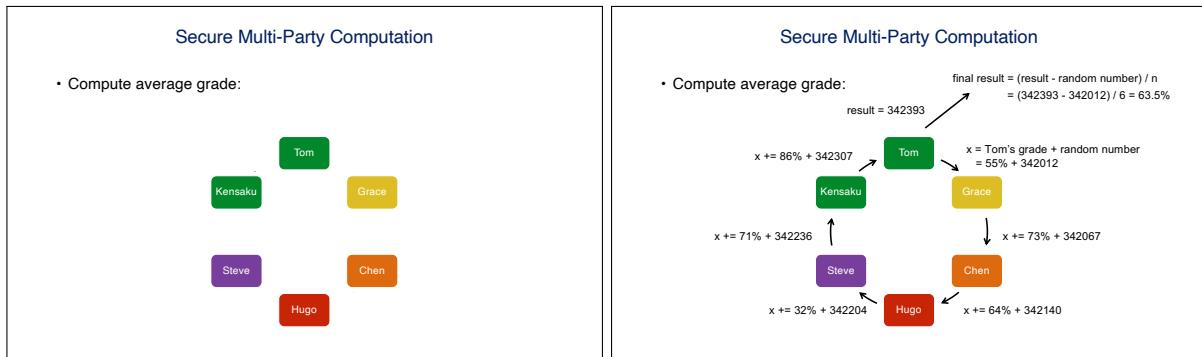
(a) There is an alternative. If you have a secret value of 5, you can split it into two parts (you don't tell anyone how you split it)

(b) We send it to clients



(c) the clients perform some computation on their share of the data and return the answer back to you.

Figure 18: You are then able to reconstruct the answer because you know how you split it. The client never saw the original value.

**Figure 19**

1.8 What is Privacy?

Anonymization

- The most straightforward approach is anonymization where identifying information is removed.
 - For example, a name may be removed from a medical record.
- Unfortunately, anonymization is rarely sufficient to protect privacy as the remaining information can be uniquely identifying.
 - For instance, given the gender, postal code, age, ethnicity and height, it may be possible to identify someone uniquely, even in a very large database.

Figure 20: In a medical dataset, a dataset is anonymized if it doesn't contain identifiable data. However, in reality it is not enough. Record linkage attacks are possible to reverse engineer.

1.8.1 k-anonymity

k-anonymity

- One approach to prevent linkage attacks is *k*-anonymity.
 - A dataset is said to be *k* anonymous if, for any person's record in a dataset, there are at least $k - 1$ other records which are indistinguishable.
 - So if a dataset is *k*-anonymous, then the best a linkage attack could ever do is identify a group of *k* records which could belong to the person of interest.
 - Even if a dataset isn't inherently *k*-anonymous, it could be made so by removing entire fields of data (like names and addresses) and selectively censoring fields of individual people who are particularly unique.

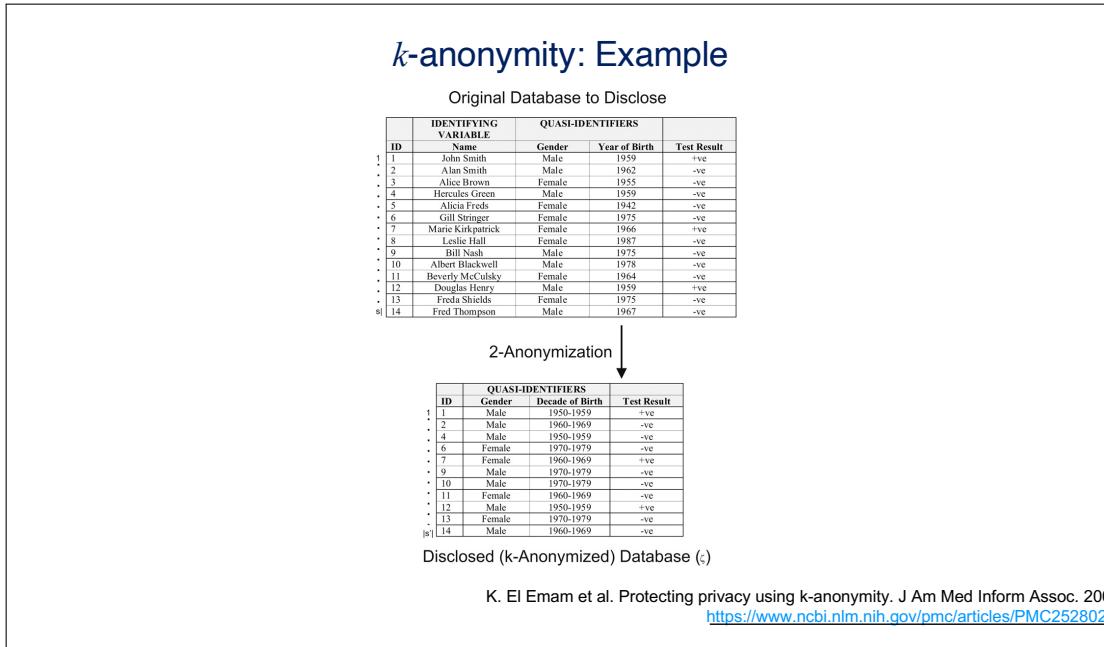


Figure 21: You can anonymize this dataset by removing the name, but the date of birth gives us a high chance of re=identifying someone.

***k*-anonymity**

Anonymization and linkage attacks

- Famous example:
 - After an insurance group released health records which had been stripped of personal information like patient name and address, a CS student was able to “deanonymize” which records belonged to politicians (including the Governor of Massachusetts) by cross referencing with public voter registers.
 - This is an example of a **linkage attack**, where connections to other sources of information work to deanonymize a dataset.
 - Linkage attacks have been successful on a range of anonymized datasets including the Netflix challenge and genome data.

 TECHNICA
POLICY —
“Anonymized” data really isn’t—and here’s why not
Companies continue to store and sometimes release vast databases of ...
NATE ANDERSON · 9/8/2009, 1:25 PM
41
The Massachusetts Group Insurance Commission had a bright idea back in the mid-1990s—it decided to release “anonymized” data on state employees that showed every single hospital visit. The goal was to help researchers, and the state spent time removing all obvious identifiers such as name, address, and Social Security number. But a graduate student in computer science saw a chance to make a point about the limits of anonymization. But it didn't prove difficult. Law professor Paul Ohm describes Sweeney's work.
<https://arstechnica.com/tech-policy/2009/09/your-secrets-live-online-in-databases-of-ruin/>

Figure 22: This doesn't work in practice as it doesn't make any assumptions about what side information you have. Here, we used additional informatino even in the public domain to re-identify.

- Unfortunately, *k*-anonymity isn't sufficient for anything but very large datasets with only small numbers of simple fields for each record.
- Intuitively, the more fields and the more possible entries there are in those fields, the more unique a record can be and the harder it is to ensure that there are *k* equivalent records.

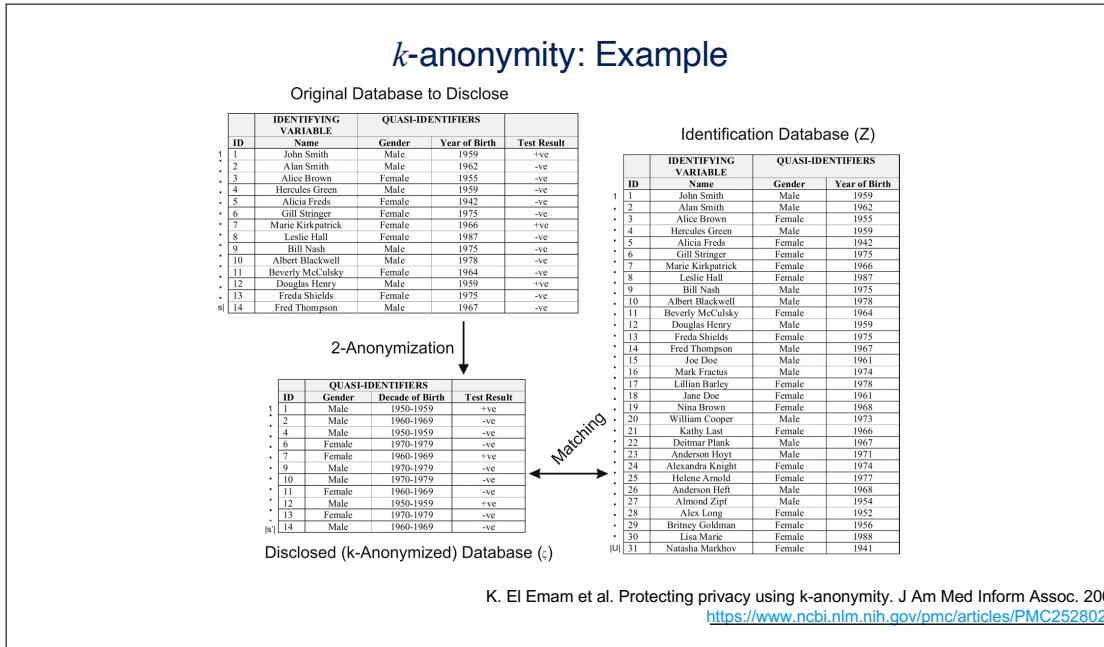


Figure 23: Here is an example; this relies on you having access to another dataset and match for example with this other dataset individual patients and to work out what their identity is.

1.9 Privacy attacks

1.9.1 Model inversion

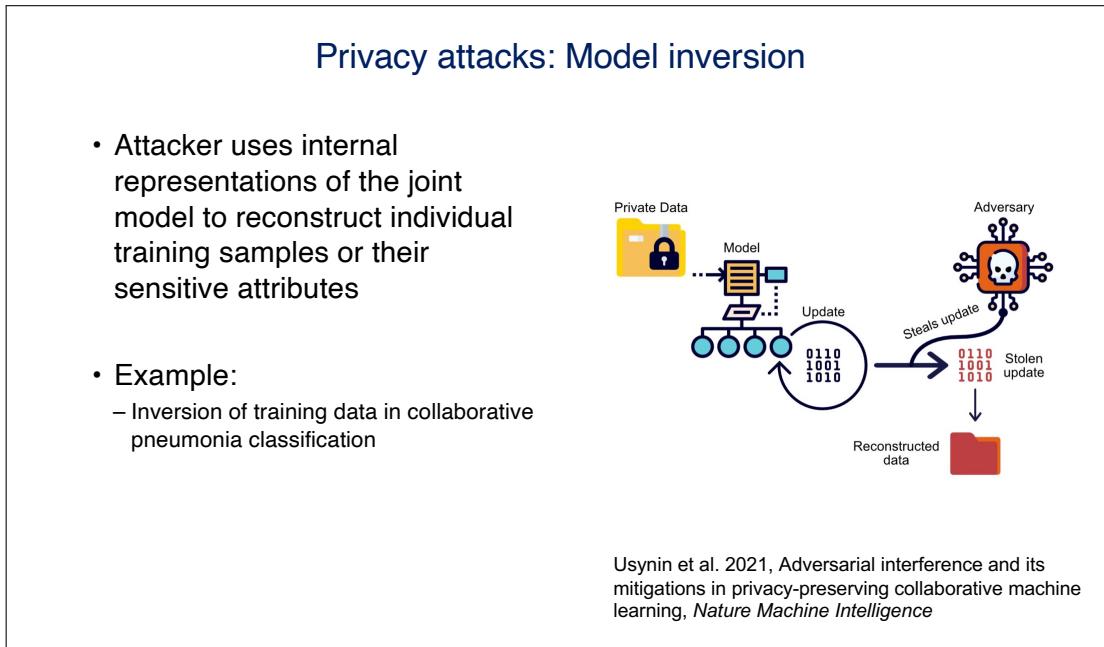


Figure 24: In the context of deeplearning, you worry most about 3 different pricay attacks you can do with images. The first attack is to try and invert the model to get the training samples.

1.9.2 Membership inference

Privacy attacks: Membership inference

- Attacker obtains a data record and determines if it was used to train a particular model
- Example:
 - Determining if a specific patient was part of the HIV-positive dataset

Usynin et al. 2021, Adversarial interference and its mitigations in privacy-preserving collaborative machine learning, *Nature Machine Intelligence*

Figure 25: Second type tries to work out if a particular dataset is in was used in the training set.

1.9.3 Attribute inference

Privacy attacks: Attribute inference

- Attacker uses model access and auxiliary information about the victim to obtain the sensitive values of their data
- Example:
 - Given access to a model trained on patient records and a specific patient's public information infer their HIV status

Usynin et al. 2021, Adversarial interference and its mitigations in privacy-preserving collaborative machine learning, *Nature Machine Intelligence*

Figure 26: Finally, don't reconstruct entire data but only some sensitive data or informatino about the data. can be common problem.

1.10 Differential Privacy

These attacks are preventable

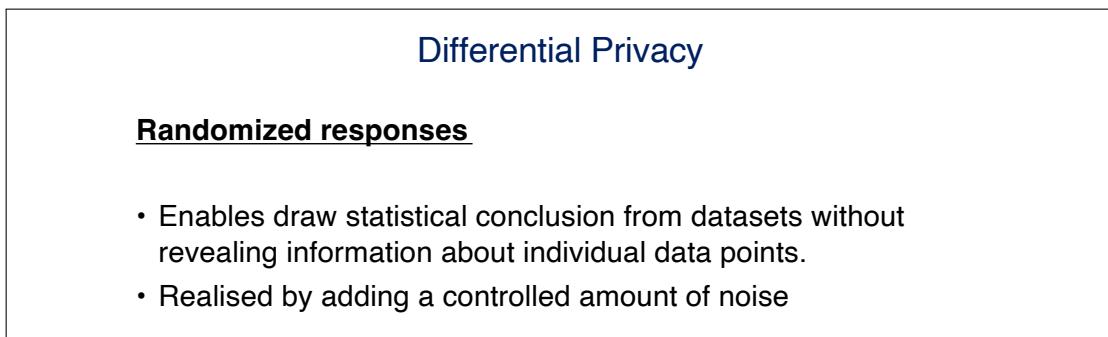


Figure 27: You randomize certain parts of the data without randomizing it enough where you destroy the data.

1.10.1 Randomized responses

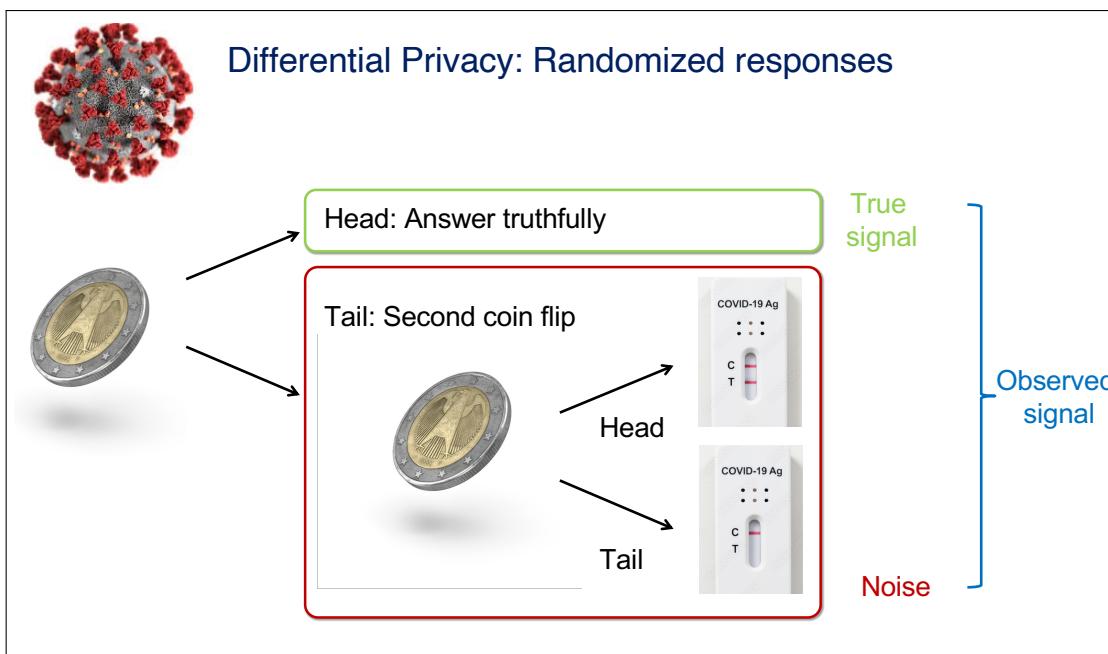


Figure 28: You want to find out who has coronavirus. But you don't really want to admit it. Still, you want some statistics. Therefore, each flip a coin and don't tell you the result of the coinflip. Following the flow above, you tell the answer. This produces a noisy answer for those that had tails. This plausible deniability means that you can't make any assessments on the individual level. But if you count the number of votes that are positive and negative, you can still calculate the prevalence.

Differential Privacy: Randomized responses

- In this example, there is a parameter which is the probability that the true response is recorded:
 - If it's very likely that the true response is recorded, then there is less privacy protection.
 - Conversely, if it's unlikely that the true response is recorded, then there is more.
 - It's also clear that, regardless of the probability, if an individual is surveyed multiple times, then there will be less protection, even if their answer is potentially randomized every time.
- Differential privacy formalizes how we define, measure and track the privacy protection afforded to an individual as functions of factors like randomization probabilities and number of times surveyed.

Figure 29: This can be applied to any machine learning model

1.10.2 Practical Application of Differential Privacy

Differential privacy

- An algorithm can be made approximately invariant to inclusion of a single data sample
- This is achieved through the addition algorithm

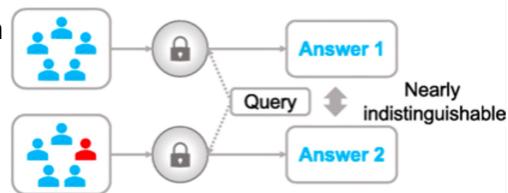


Figure 30: even if you leave one person out you want the model to give virtually the same answer. This is a good thing because you don't know which person was left out and you can't infer anything about the person. "This is nearly indistinguishable. You have a parameter ϵ which tells you how strict the privacy guarantees are. For strict privacy guarantees (small epsilon) with small sample size this won't work, so you'll have to add so much noise that the output of the model utility will drop down to zero."

Differential privacy (continued)

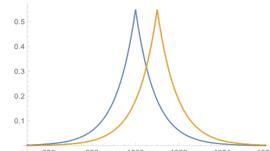
- This can be mathematically defined using the following (relaxed) expression:

A process A is ϵ -differentially private if for all databases D_1 and D_2 which differ in only one individual:

$$\mathbb{P}[A(D_1) = O] \leq e^\epsilon \cdot \mathbb{P}[A(D_2) = O]$$

Differential privacy (continued)

- This is done through the addition of noise to the output of the query so then the results in two datasets look approximately like this:



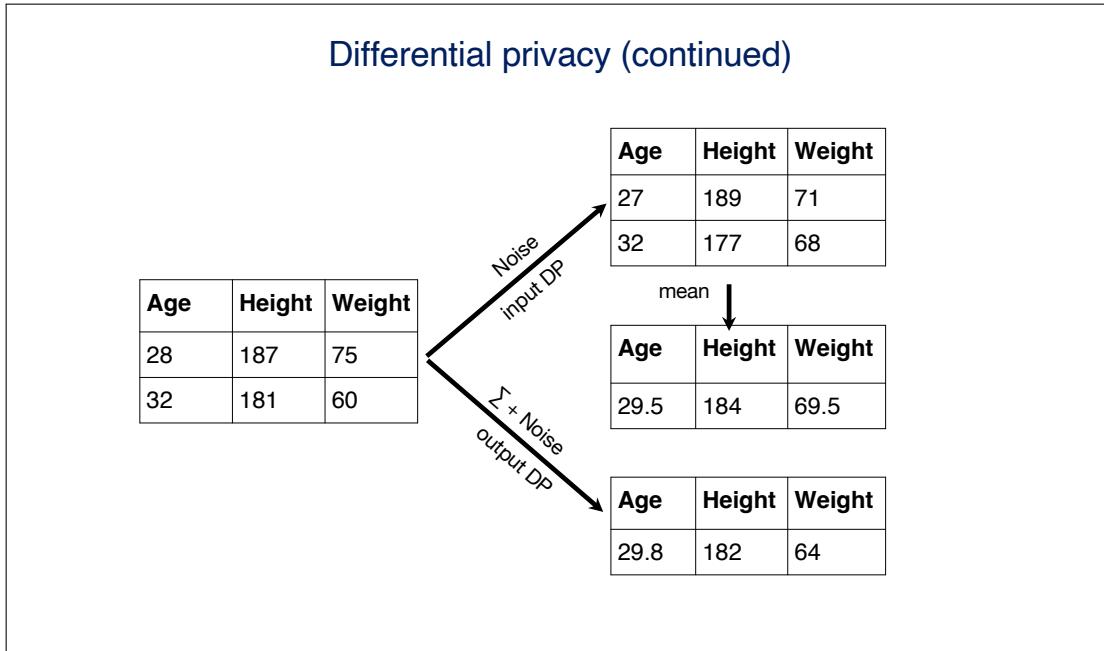


Figure 31: If you do this with tabular data you can add randomized noise to each entry. You can do this with input data or output data. With images, you do it with the input.

1.10.3 Concrete Mitigation: stochastic gradient descent (DP-SGD)

Concrete mitigation: Differentially private stochastic gradient descent (DP-SGD)

1. Compute gradients for each individual sample (they represent independent clients)
2. Clip the calculated gradients to obtain a known sensitivity
3. Add the noise scaled by the sensitivity from step 2
4. Perform the gradient descent step

Abadi, Martin, et al. "Deep learning with differential privacy." *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016.

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \dots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t , noise scale σ , group size L , gradient norm bound C .

Initialize θ_0 randomly

for $t \in [T]$ do

- Take a random sample L_t with sampling probability L/N
- Compute gradient**

 - For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, x_i)$
 - Clip gradient**

 - $\tilde{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C})$
 - Add noise**

 - $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L} \sum_i (\tilde{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$
 - Descent**

 - $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Figure 32: It performs one extra step where you compute the gradient where you clip your gradeint and add noise. Clipping limits the norm of each gradient; limiting the influence that each training example can have on your overall gradient. Since noise is noramlly distributed this should average itsself out. This completely destroyed the model's ability to reconstruct training data!

When you force deep learning to not force the model to memorize the data, you're instead normalizing and preventing overfitting, since often regualirsation includes adding noise.

2 Interpretability and Explainability

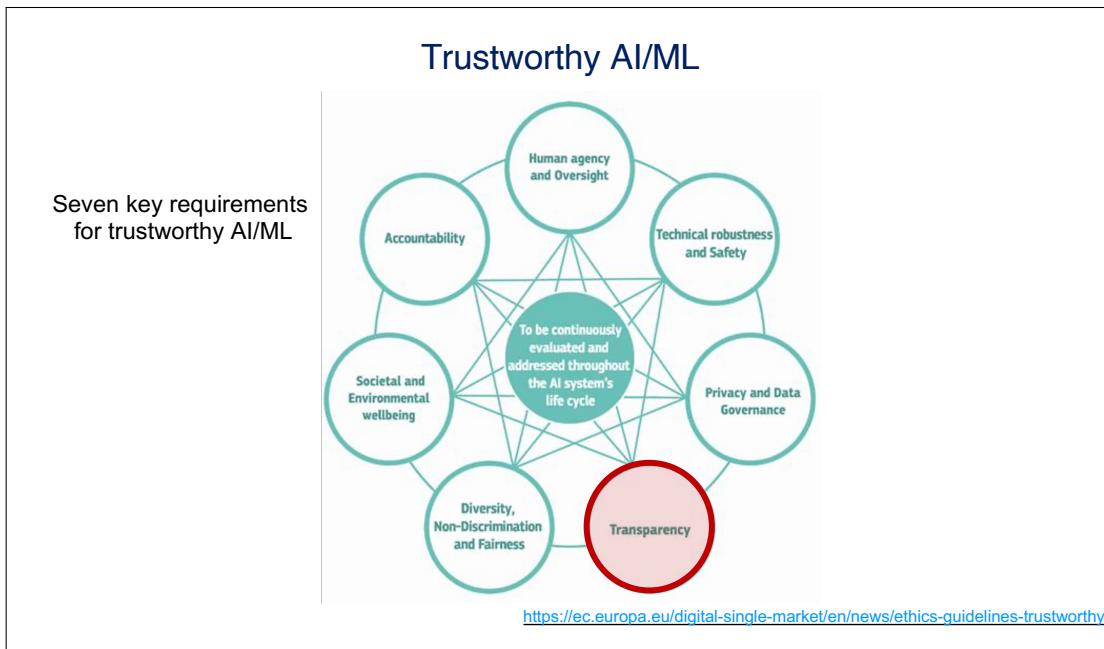


Figure 33: caption

2.1 Interpretability

2.1.1 Why is it important?

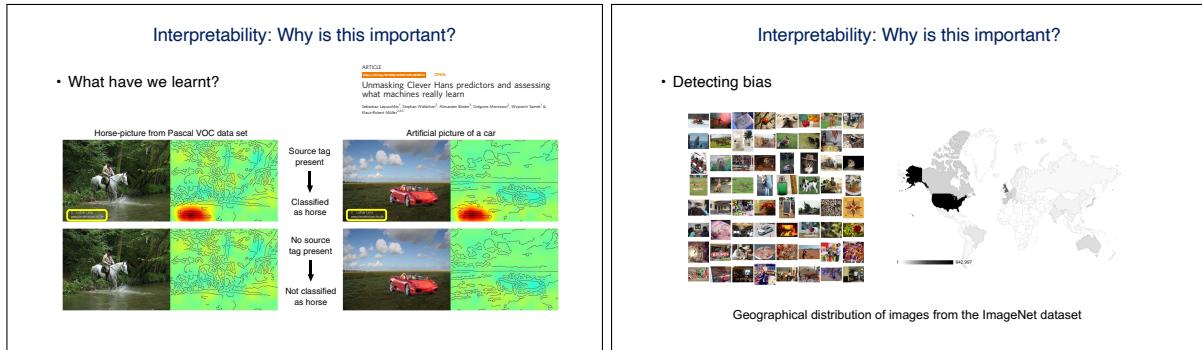
Interpretability: Why is this important?

- This is not a new problem, so why now?
 - Complexity and prevalence!
 - Safety is critical
 - Generating knowledge
- Debugging machine learning
 - Why does my model not train?
 - Why does my model exhibit unexpected/unintuitive behaviour?
- To use machine learning responsibly we need to ensure that
 - Our values are aligned
 - Our knowledge is reflected

Figure 34: Useful to gain insights into what your machine learning does. Furthermore, it is useful to use as a debugging tool

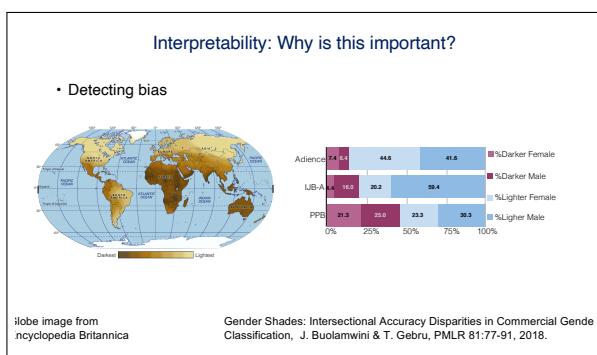


(b) a classifier might not work on the right, in some cases, it might not be because of the training data.

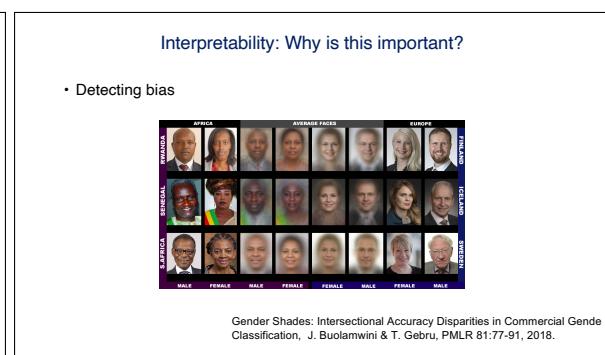


(c) Here, a model is trained to predict whether there is a horse or car in the picture. Some of the data in the other countries dataset had text. This text tells you a lot about the image. It might not be obvious nor apparent, but an interpretable method allows you to query this model and extract which features in the image lead to the classification

(d) Data distribution amongst countries may not work in



(e) This also is bad with face detection for skin colour



(f) It can be useful for detecting bias in the models

2.1.2 Common misunderstandings

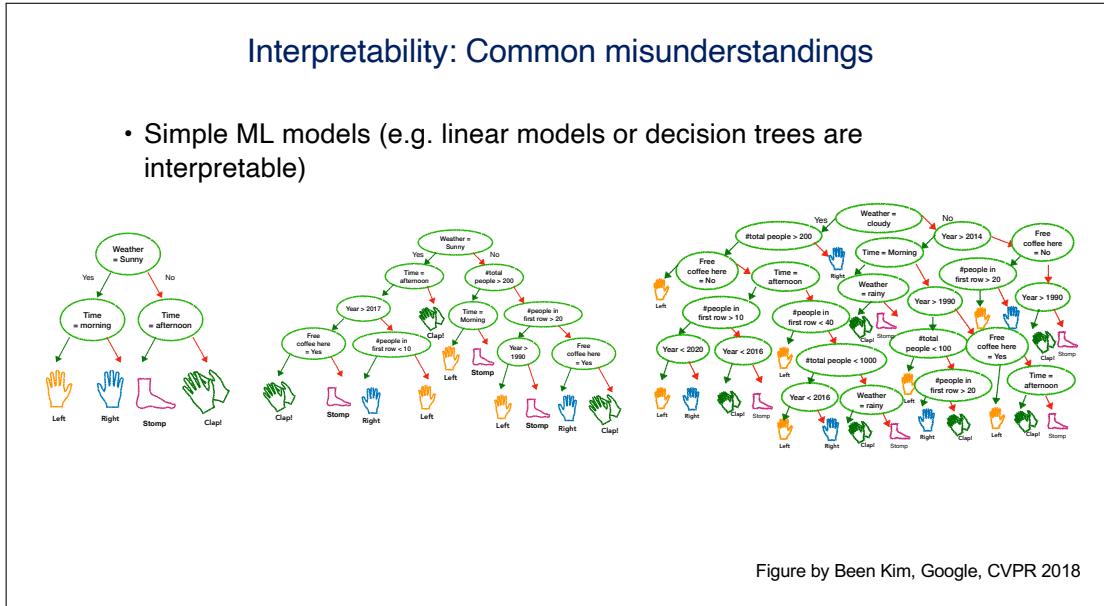


Figure 35: decision trees are interpretable, but these can get very complicated and no longer interpretable.

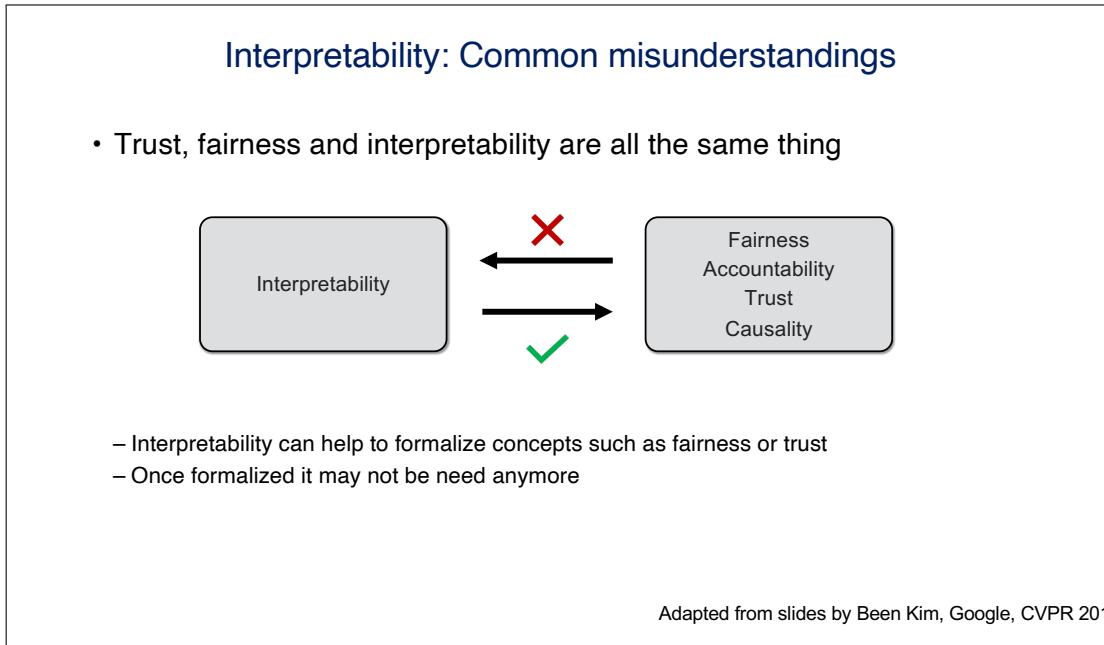


Figure 36: If you want to trust a method, you need it to be interpretable. But because a method is trustworthy, it doesn't mean it is interpretable.

Interpretability: Common misunderstandings

- We don't always care about interpretability:
 - No significant consequences or when predictions are all you need
 - Sufficiently well-studied problem
 - Prevent gaming the system

Figure 37: You also sometimes don't care about interpretability. Examples above. Gaming the system means we can trick the system if we know how it works.

2.2 How can we interpret an existing ML model?

How can we interpret an existing ML model?

- Ablation test: How important is a data point or feature?
 - Train without certain data or features and observe/study the impact
 - Difficult and expensive

Figure by Been Kim, Google, CVPR 201

How can we interpret an existing ML model?

- Ablation test: How important is a data point or feature?
 - Train without certain data or features and observe/study the impact
 - Difficult and expensive

Figure by Been Kim, Google, CVPR 201

(a) One possible way is to say which feature is really important here? The ablation test removes a feature

(b) this effectively projects it onto feature no.1 and thus is harder to classify because of the large overlap.

Figure 38: This doesn't scale well

How can we interpret an existing ML model?

- Fit functions (use first derivatives)
 - Sensitivity analysis
 - Saliency maps

Figure by Been Kim, Google, CVPR 201

How can we interpret an existing ML model?

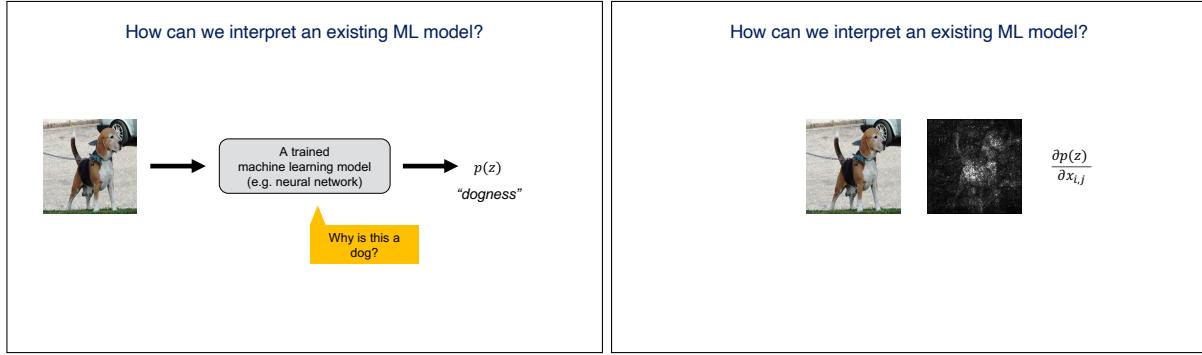
- Fit functions (use first derivatives)
 - Sensitivity analysis
 - Saliency maps

Figure by Been Kim, Google, CVPR 201

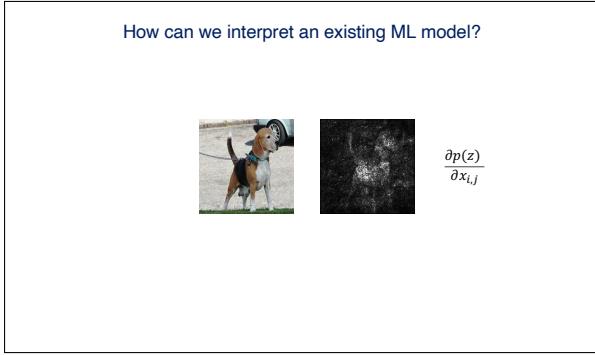
(a) Since we work with images, we can do sensitivity analysis or saliency maps. We take a particular sample and look at the gradient of the sample w.r.t feature1 and feature 2. This tells you how drastically if you change the feature value how the classification will change.

(b) If you do this on another value nearby you might get a completely different answer which may give very contradictory explanations.

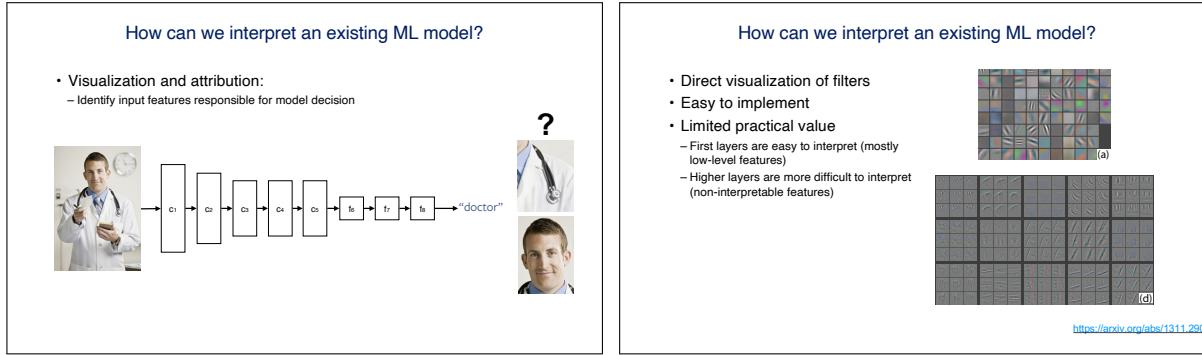
22



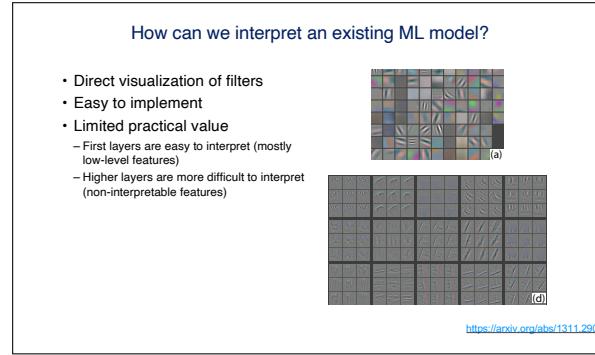
(c) A classifier predicts how much it looks like a dog. We want it to give an explanation as to why its a dog.



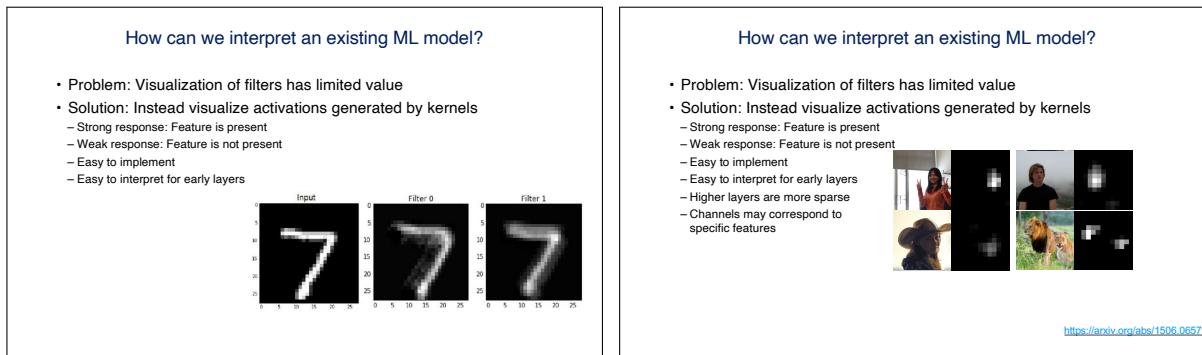
(d) In this example, sensitivity analysis gets an output like this. We compute the gradient on the output function w.r.t the input pixels (because these are teh features).



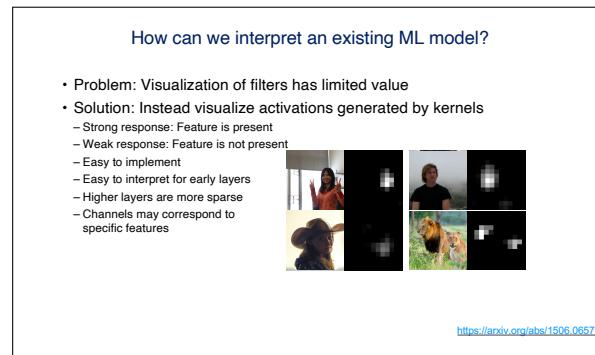
(e) With the stereotypical doctor image, i want it to re-
conzie reasons. Here, the top image is better. For example, input pixels. Instead, we can visaulise the convolutional
the model may take a shortcut and predict that a man is features. However, this is not interpretable
more likley to be a doctor.



(f) How do we get to this? We can differentiate w.r.t the features. However, this is not interpretable



(g) A more practical approach is to focus on occlusion maps. We can give it a sample input and see what it outputs.



2.2.1 Occlusions

How can we interpret an existing ML model? Occlusions

- Idea: Mask out region in the input image and observe network output
- If masked out region causes a significant drop in confidence, the masked-out region is important

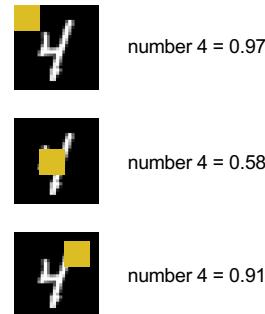
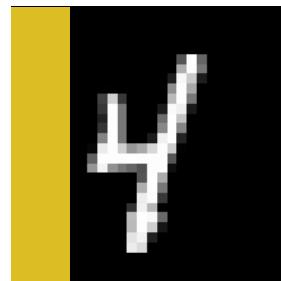
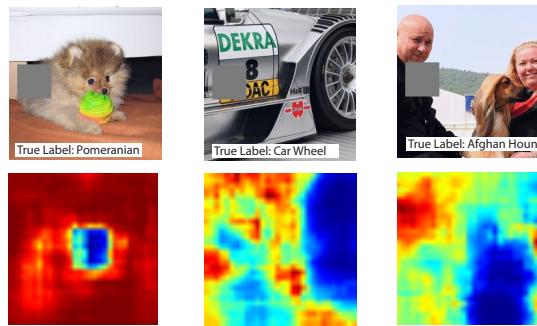


Figure 39: Here, slide an occlusion filter among the MNIST data. Here the model reacts more sensitively to some locations than others. To visualise this, we use a Saliency Map

How can we interpret an existing ML model? Occlusions

- Idea: Mask out region in the input image and observe network output
- If masked out region causes a significant drop in confidence, the masked-out region is important



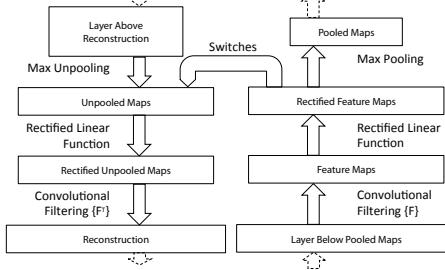
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, K. Simonyan, A. Vedaldi, and A. Zisserman ICLR, 2014.

2.2.2 Saliency maps

How can we interpret an existing ML model? Saliency maps

- **DeconvNet**

- Given a trained network and an image
- Choose activation at one layer (set all others to zero)
- Invert network



- No training involved
- Backward pass in network is almost identical to backpropagation (apart from ReLUs)

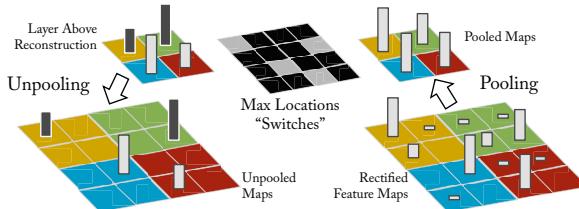
Visualizing and Understanding Convolutional Networks, M. D. Zeiler and R. Fergus, ECCV 2011

Figure 40: What this approach did was try to invert the process of the forward pass. What you do now, is you take an image as input, you put it through the network up until a layer, then try to reverse it.

How can we interpret an existing ML model? Saliency maps

- **DeconvNet**

- Given a trained network and an image
- Choose activation at one layer (set all others to zero)
- Invert network



- No training involved
- Backward pass in network is almost identical to backpropagation (apart from ReLUs)

Visualizing and Understanding Convolutional Networks, M. D. Zeiler and R. Fergus, ECCV 2011

Figure 41: for each operation you need an inverse information (difficult since CNNs contract information). You do something similar as backpropagations, but in the forward pass you record information for the maximum locations. Then when you do back propagation you slap the values into where the max of the forward pass was.

**How can we interpret an existing ML model?
Saliency maps**

- **DeconvNet**

Note the exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs

Visualizing and Understanding Convolutional Networks, M. D. Zeiler and R. Fergus, ECCV 2011

Figure 42: Here, there are many activations where the dogs face was activated. This allows you to investigate what each layer does.

**How can we interpret an existing ML model?
Saliency maps**

- **Question:**
 - Which pixels are most significant to a neuron?
 - How would they need to change to most affect the activation of the neuron?
- **Solution:**
 - Use back propagation but differentiate activation with respect to **input pixels, not weights**

weights of the network are fixed $\frac{\partial h}{\partial x_i}$ X

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, K. Simonyan, A. Vedaldi, and A. Zisserman ICLR, 2014.

Figure 43: backpropagate with respect to input pixels (not the weights). But this isn't much more useful than just localising where the point of interest is.

2.2.3 Saliency maps - Gradient (backpropagation)

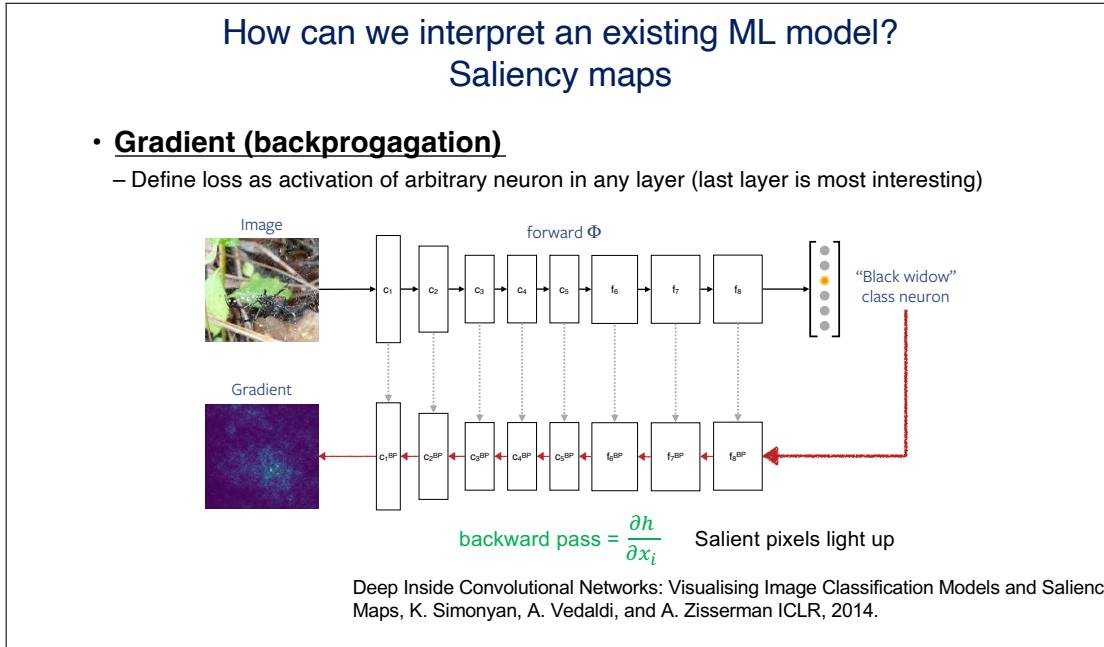


Figure 44: Simplest way is to do backprop: do the max activation and pretend that only this is activated and do your backpropagation

2.2.4 Saliency maps - Guided backpropagation

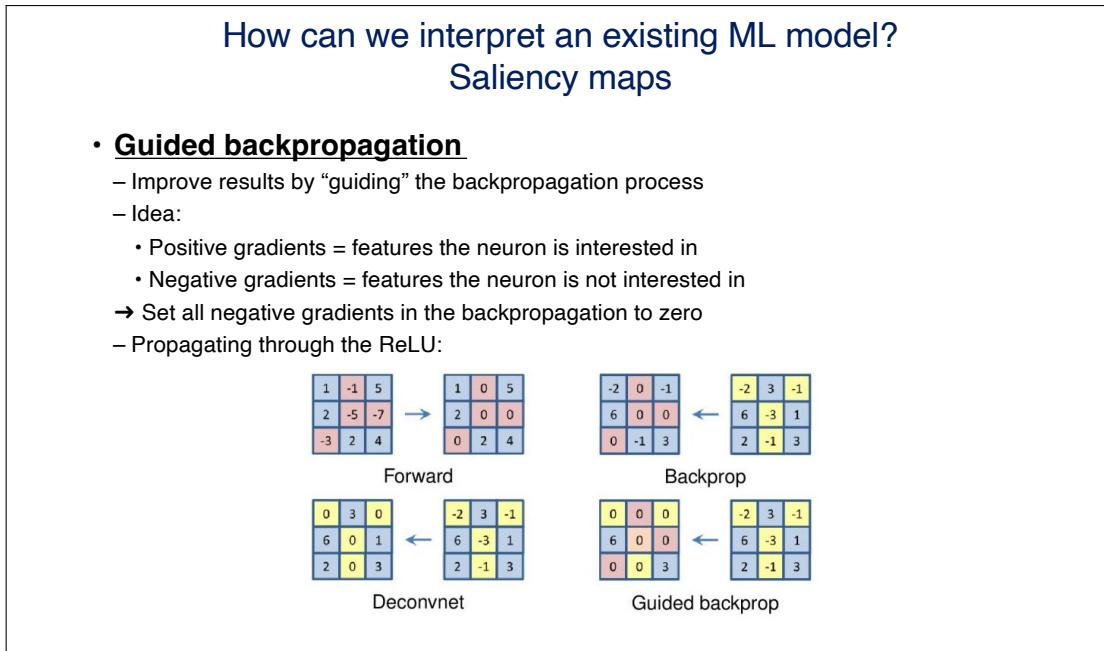


Figure 45: This makes a simple modification; we are not interested in neurons that give negative gradients because negative gradients don't contribute anything to the output. The positive outputs represent features the neuron is interested in. This way, you set all the negative weights to zero.

Below, you see the three different schemes, standard backprop, deconvnet and guided backprop. What they differ in is how they deal with negative values.

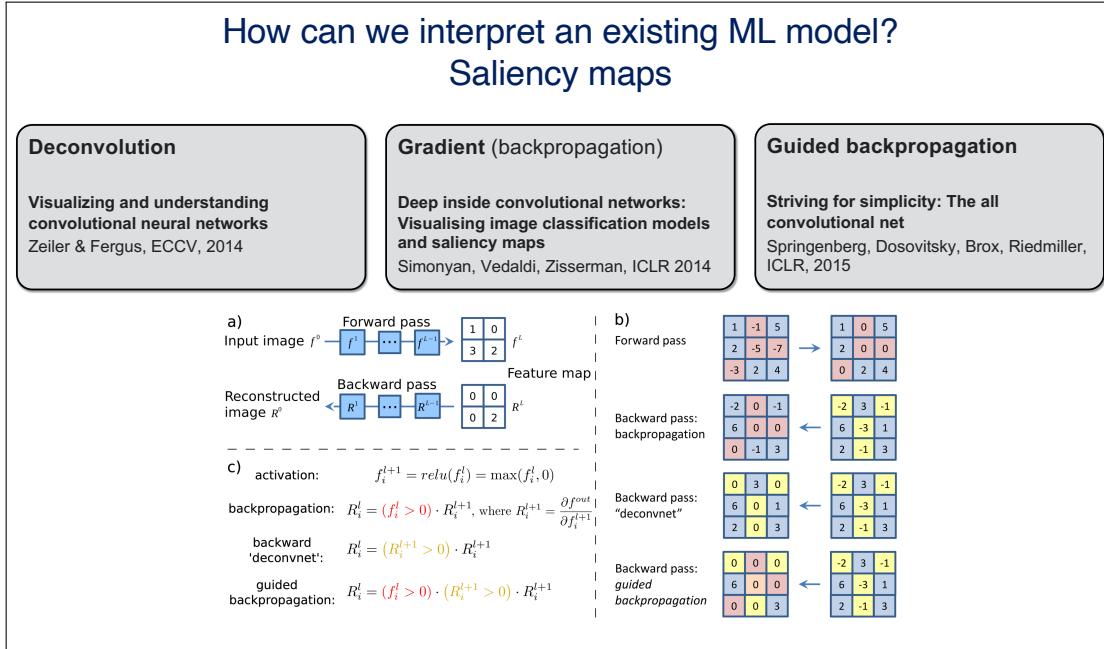


Figure 46: Rules for computing by hand

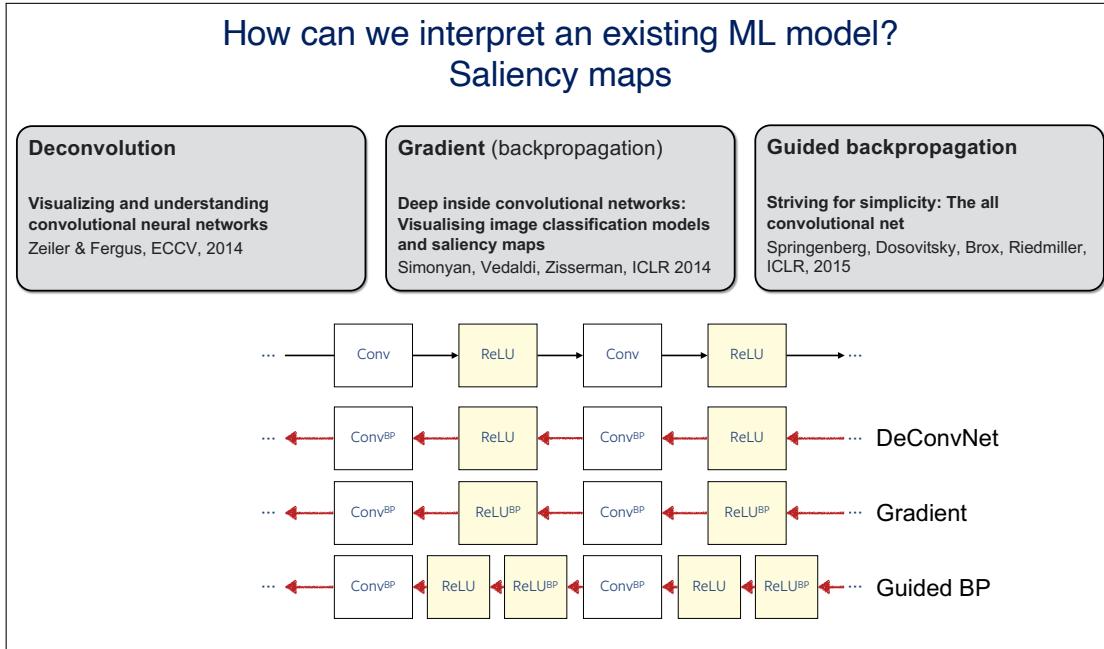


Figure 47

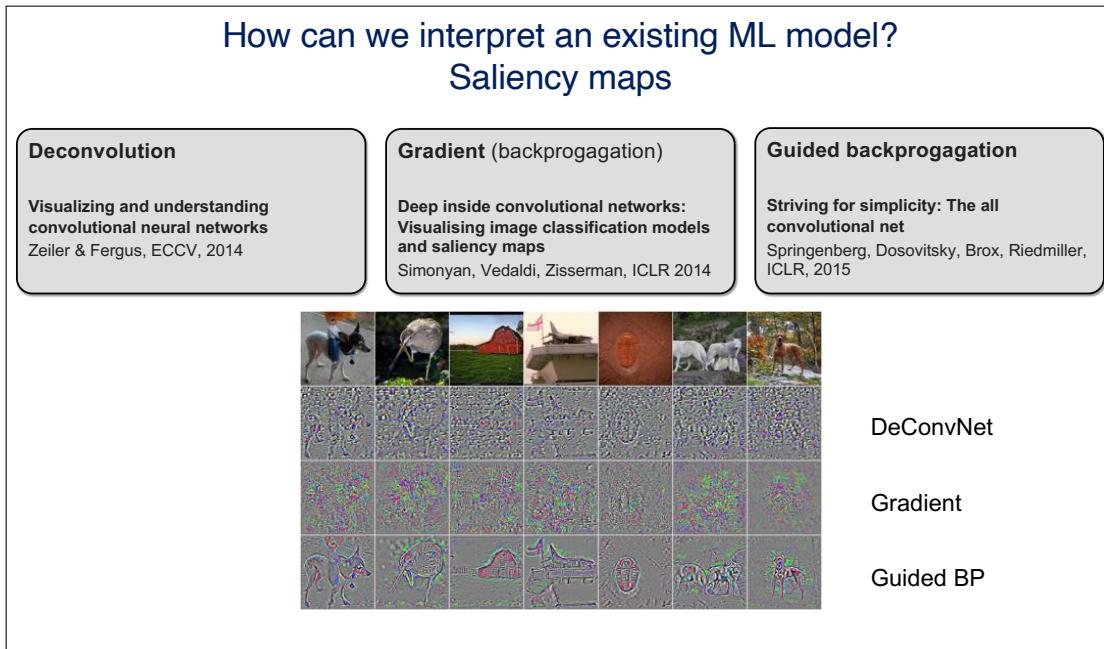


Figure 48: Guided backprob gives most meaningful results usually

2.3 Cam and Grad-Cam

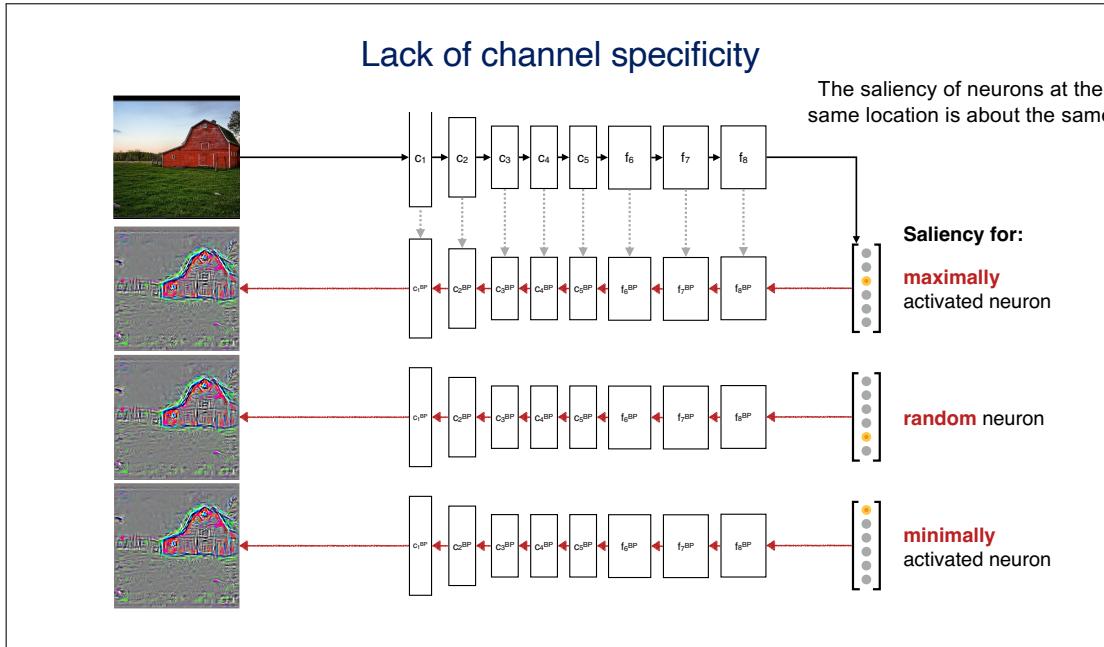


Figure 49: The saliency map should also change depending on the output of the network if you have a saliency map for maximally activated neuron and minimally activated neuron and they give the same map that doesn't make sense.

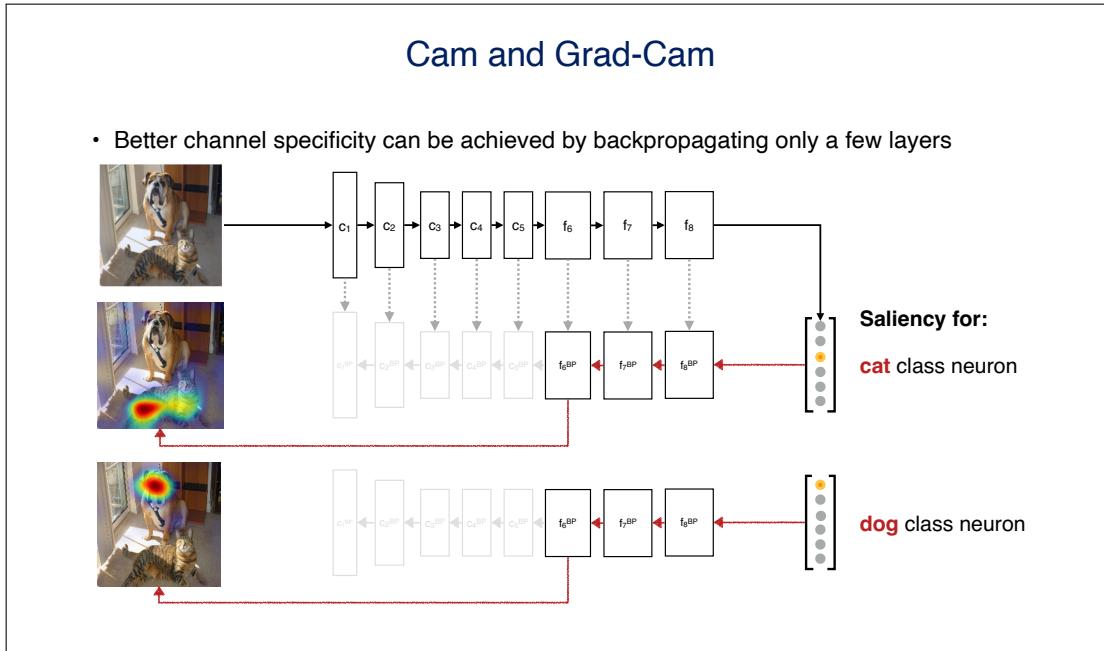
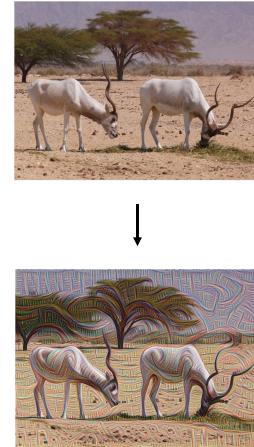


Figure 50: The Grad-Cam method allows you to query in much more detail what the different input channels are doing in terms of activation

2.4 DeepDream / Inceptionism

DeepDream / Inceptionism

- Attempt to understand the inner workings of the network
- Optimize with respect to image
- Idea
 - Arbitrary image or noise as input
 - Instead of adjusting network parameters, tweak image towards high “X” where “X” can be
 - Neuron/Activation map/Layer
 - Logits/Class probability
 - Search for images that are “interesting”
 - Different layers enhance different features



<https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.htm>

Figure 51: They use ideas of inverse problems to see what the networks are learning

DeepDream / Inceptionism

- Find an image x such that the activation $\phi_n(x)$ at layer n is high

$$\max_x \phi_n(x) - \lambda \mathcal{R}(x)$$

Regularizer: e.g. L1 or L2 norm of image



<https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.htm>

Figure 52: we want to find an image that has the same activation that you have already observed for an existing input image and you try to reverse engineer this.

2.5 Inversion

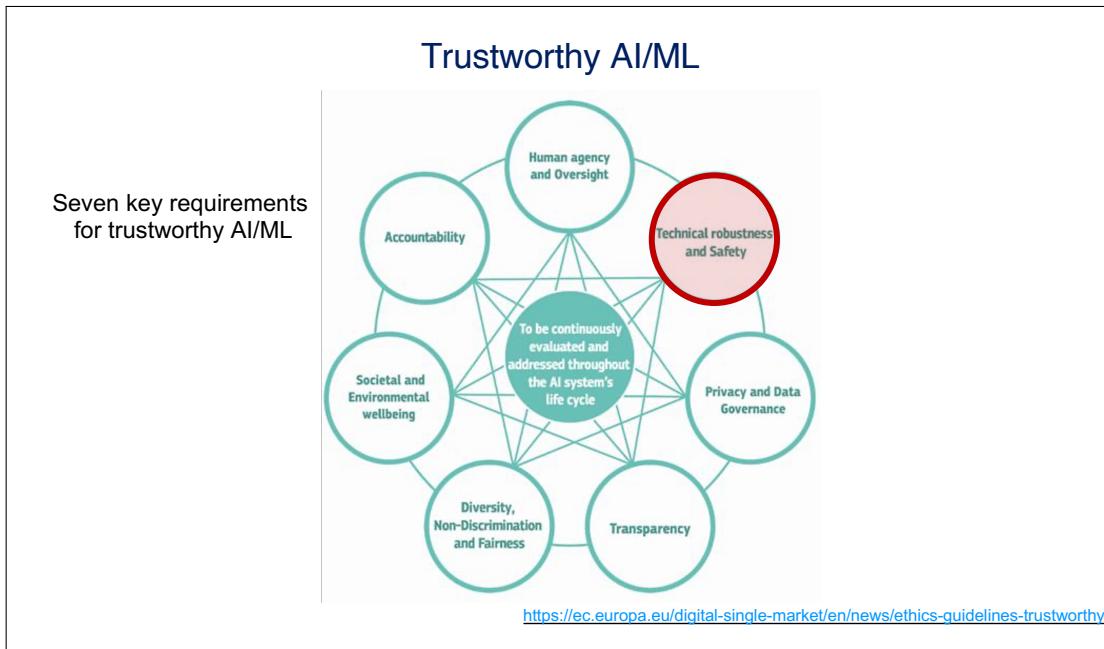
Inversion

- Inversion attempts to construct an image from a layer activation \hat{y} :

$$\hat{x} = \min_x (\|\phi(x) - \hat{y}\|_2^2 + \lambda \mathcal{R}(x))$$

- \hat{x} is the reconstructed image
- $\phi(x)$ is the network output for input image x
- \hat{y} is the desired activation
- \mathcal{R} is the regularizer

3 Robustness: Adversarial Methods



3.1 Adversarial Attacks

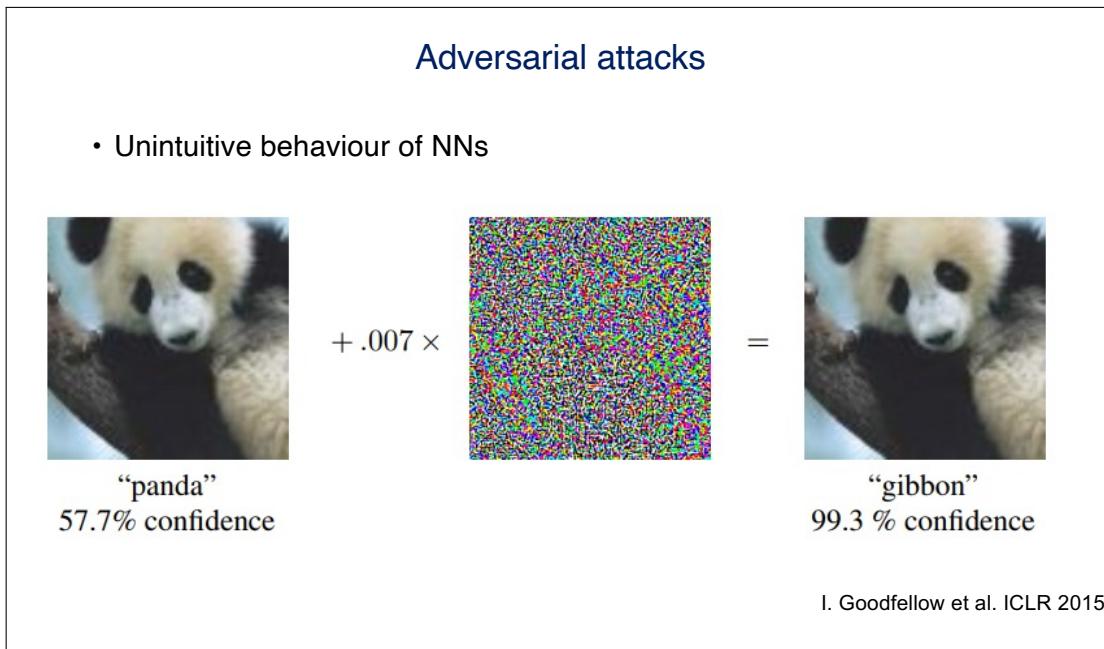


Figure 53: Two images have been produced; the difference between the two image is a small scaling factor of noise. The small difference shouldn't be enough to change the classification.

3.1.1 Perturbation

Adversarial attacks - Perturbation

- Assume a linear classifier:

$$\theta^T x$$

- We can think of an adversarial example that contains a small, non-perceivable perturbation to the input. Let's denote the perturbation as η :

$$\tilde{x} = x + \eta$$

- Then, the logits of the classifier would be

$$\begin{aligned}\theta^T \tilde{x} &= \theta^T(x + \eta) \\ &= \theta^T x + \theta^T \eta\end{aligned}$$

Figure 54: We add to x another vector and see what happens; we can't guarantee that despite the noise being small, the output will be the same.

Adversarial attacks - Perturbation

- Given a small perturbation η , the effect of the perturbation on the logits of the classifier is given by $\theta^T \eta$.
- Idea:
 - Find a η that causes a change that is non-perceivable and ostensibly innocuous to the human eye, yet destructive and adverse enough for the classifier to the extent that its predictions are no longer accurate.
 - An adversarial example is one that which maximizes the value of $\theta^T \eta$ to sway the model into making a wrong prediction

Figure 55: If you want to attack a model, you want to find a small perturbation that changes the classification drastically

Adversarial attacks - Perturbation

- Problem:
 - Need a constraint on η ; otherwise, one could just apply a large perturbation to the input
- Solution:
 - Apply a constraint such that

$$\|\eta\|_\infty \leq \epsilon \quad \|x\|_\infty = \max_i \{|x_i|\}$$

- Assume a perturbation:

$$\eta = \epsilon \cdot \text{sign}(\theta)$$

- What are the bounds of this perturbation?

I. Goodfellow et al. ICLR 2015

Figure 56: This is trivially easy to do with a large perturbation. However, we constrain it such that the largest element in the vector is less than epsilon (L_{inf} norm). Then we assume the smallest perturbation where the sign function produces either 1 or -1.

Adversarial attacks - Perturbation

- What are the bounds of this perturbation?

$$\eta = \epsilon \cdot \text{sign}(\theta)$$

$$\begin{aligned} \theta^T \eta &= \epsilon \cdot \theta^T \text{sign}(\theta) \\ &= \epsilon \|\theta\|_1 \\ &= \epsilon m n \end{aligned}$$

Here the average magnitude of an element of θ is given by m

I. Goodfellow et al. ICLR 2015

Figure 57: after some math we can find the magnitude of the vector.

Adversarial attacks - Perturbation

- This means that the change in activation given by the perturbation increases linearly with respect to n (or the dimensionality).
- If n is large, one can expect even a small perturbation capped at ϵ to produce a perturbation big enough to render the model susceptible to an adversarial attack.
- Remember that for images $n = \text{no. of pixels}$
- Such perturbed examples are referred to as **adversarial examples**

I. Goodfellow et al. ICLR 2015

Figure 58: What this means is that as the dimensionality increases n gets very large. So this magnitude increases drastically with the number of pixels which means it becomes easier to find noise that will attack the model.

3.1.2 Fast Gradient Sign Method

Adversarial attacks - Fast Gradient Sign Method

Key idea:

- Perform gradient descent in order to maximize the loss (the goal of adversarial attack).
- Consider the input image x to be a trainable parameter and compute the gradient with respect to the input image to create a perturbation.

$$\eta = \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(\theta, x, y))$$

- An adversarial example can be created as:

$$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(\theta, x, y))$$

I. Goodfellow et al. ICLR 2015

Figure 59: Now, you find the optimal vector that attacks the model, meaning that you add the gradient of loss function and then create this vector of zeros -1s and 1s depending on which direction the gradient points, you can create an adversarial example very easily.

3.2 Adversarial attacks

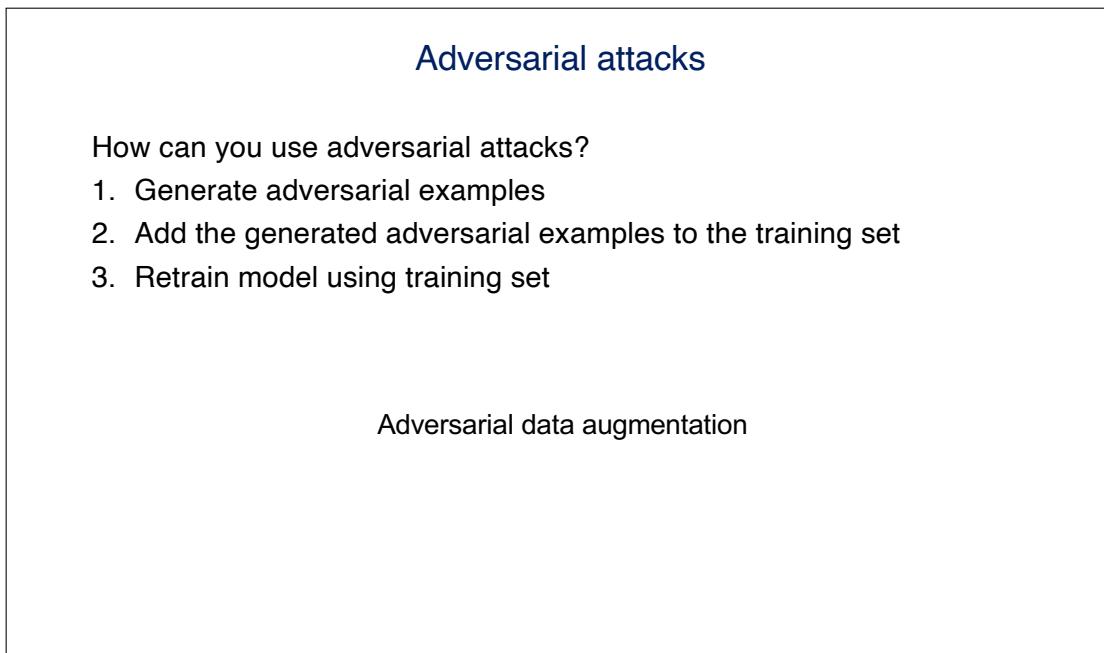


Figure 60: You can't defend that easily.

A References

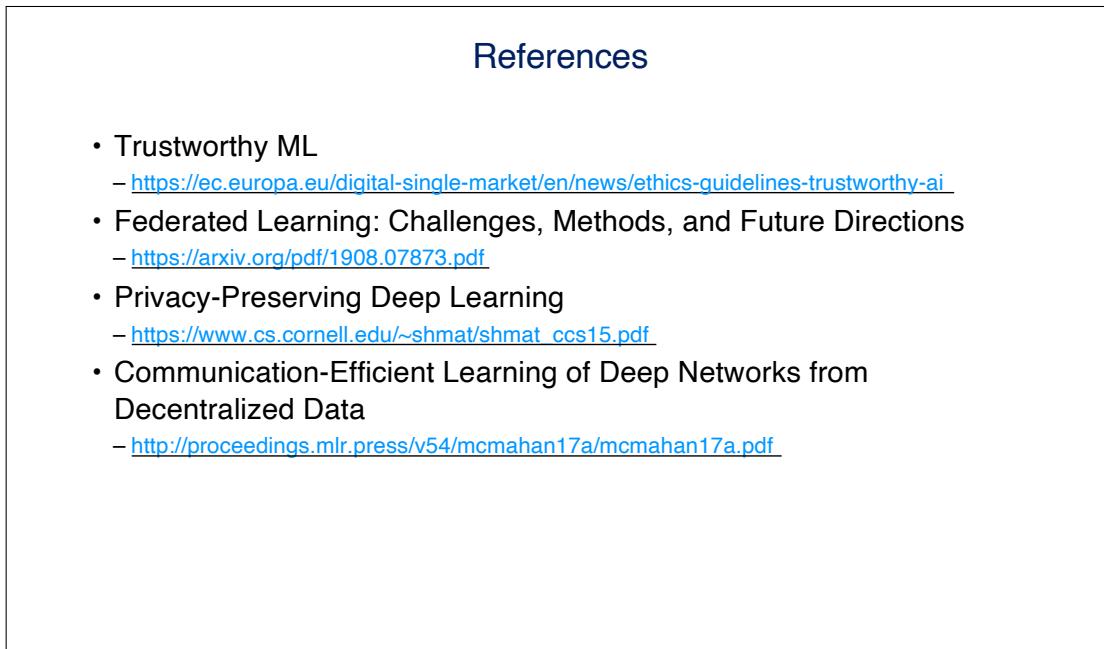


Figure 61: caption

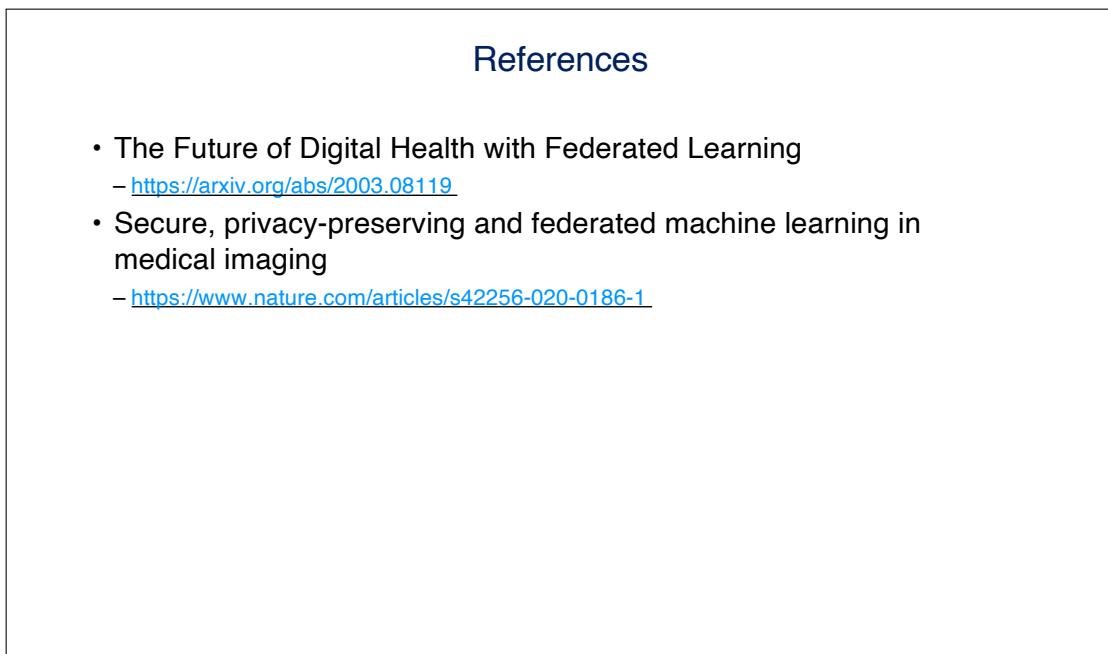


Figure 62: caption

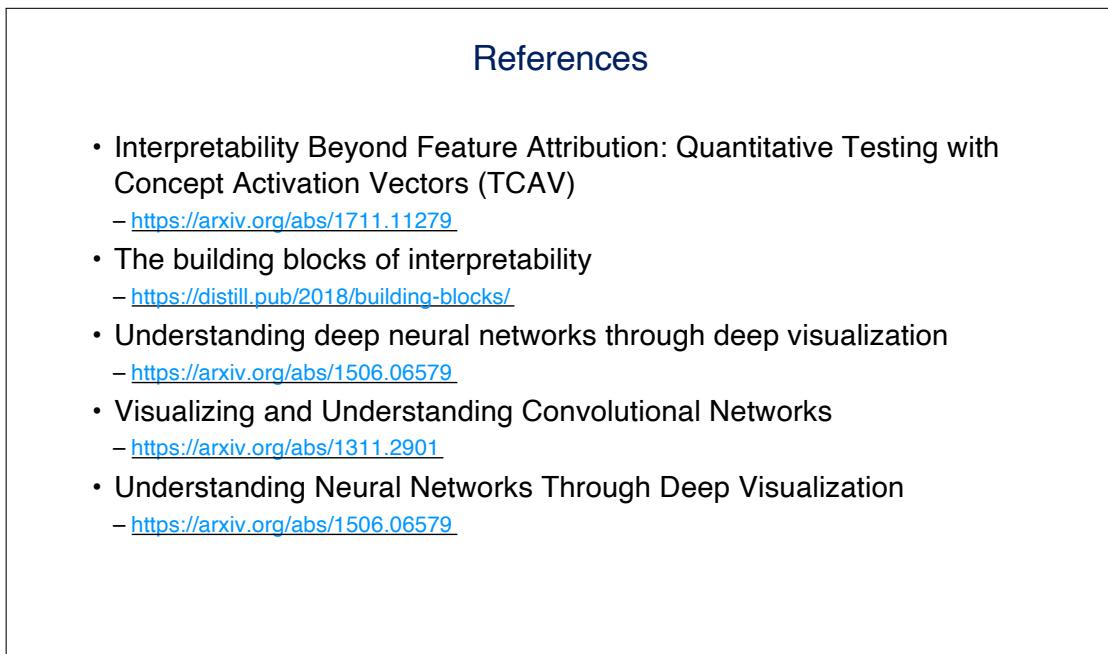


Figure 63: caption

References

- [1] Dmitrii Usynin et al. “Adversarial interference and its mitigations in privacy-preserving collaborative machine learning”. In: *Nature Machine Intelligence* 3.9 (Sept. 2021), pp. 749–758. ISSN: 2522-5839. DOI: [10.1038/s42256-021-00390-3](https://doi.org/10.1038/s42256-021-00390-3). URL: <https://doi.org/10.1038/s42256-021-00390-3>.