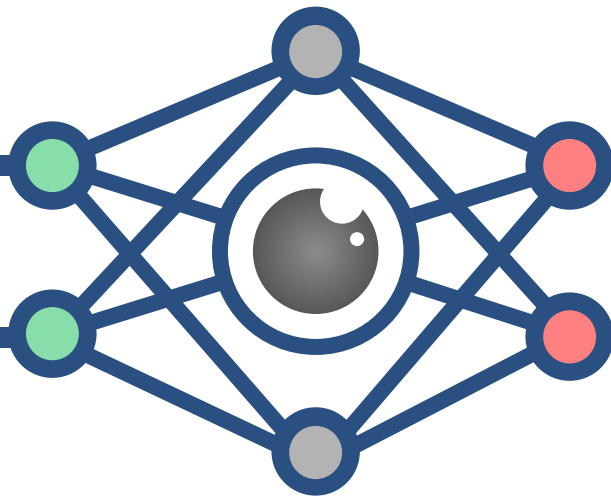


CS3485

Deep Learning for Computer Vision



Lec 10: Adversarial Examples and Self-supervision

Announcements

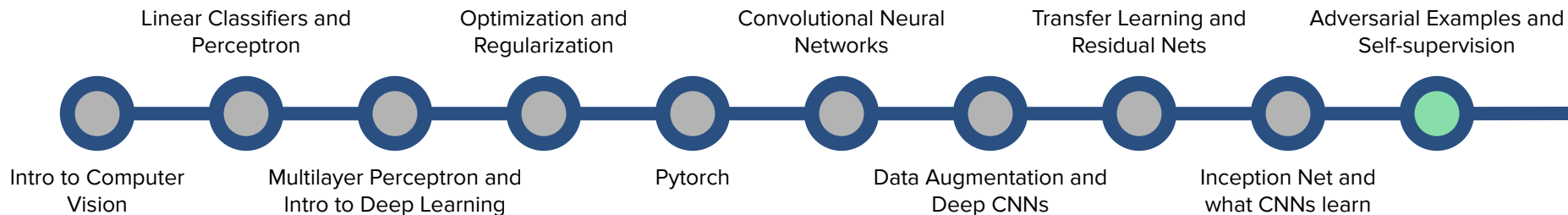
- Midterms:
 - They were graded and will be delivered back to you at the end of the class,
 - Their solution sheet and rubric is on Canvas (go to assignments, then “Midterm 1”).
 - I may have made made mistakes correcting your exams. Let me know it and I’ll fix them! I just ask to wait until next Monday to storm in.
- Logistics for the next weeks:
 - My baby is coming! The due date is Nov *1st*,
 - Some classes may go online and/or get shifted around, hopefully none of them will be cancelled!
- Colab and training deep nets:
 - Google provided us with credits for their cloud environment (GPC), but I learned they do not translate to Colab credits....
 - We are figuring out other options, and one of them is to use Bowdoin’s HPC.
 - We may skip lab next week because of that.

Announcements

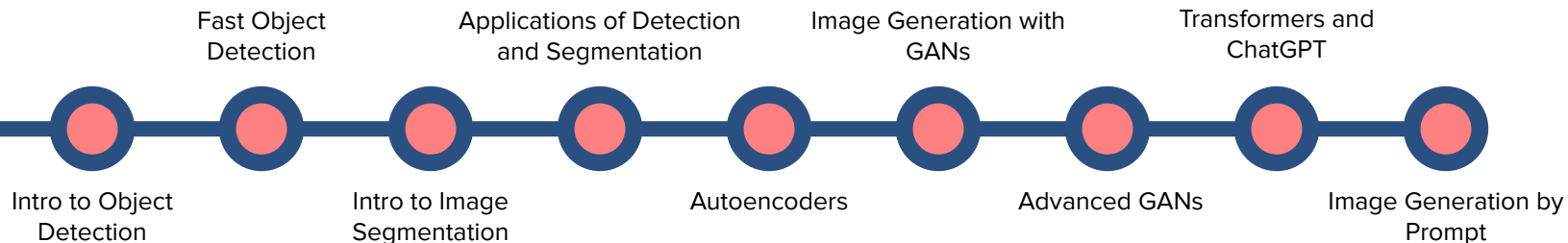
- Final project:
 - **Groups of 2-4 students.**
 - Two options for the **theme**:
 - i. Do a literature review on the SOTA of some Computer Vision task (like Image Classification for example). In this case, the review should also consider the implementations of the methods and some comparison between them.
 - ii. Try to solve any problem of your choice using Deep Learning (it does not need to be in Computer Vision, it can be involving audio, text, etc.)
 - The teams should send a **proposal** the Nov 10th with a problem statement, motivation, the main tasks and how each student will contribute to it.
 - The **presentation** will be in person on Dec 4th and/or Dec 6th, and it should last for at least 10 min, such that each student member presents for at least 4 min. It should also present some sort of **demonstration** (it can be the inference on some set of images).
 - **More information on it on the Syllabus and on the website.**

(Tentative) Lecture Roadmap

Basics of Deep Learning

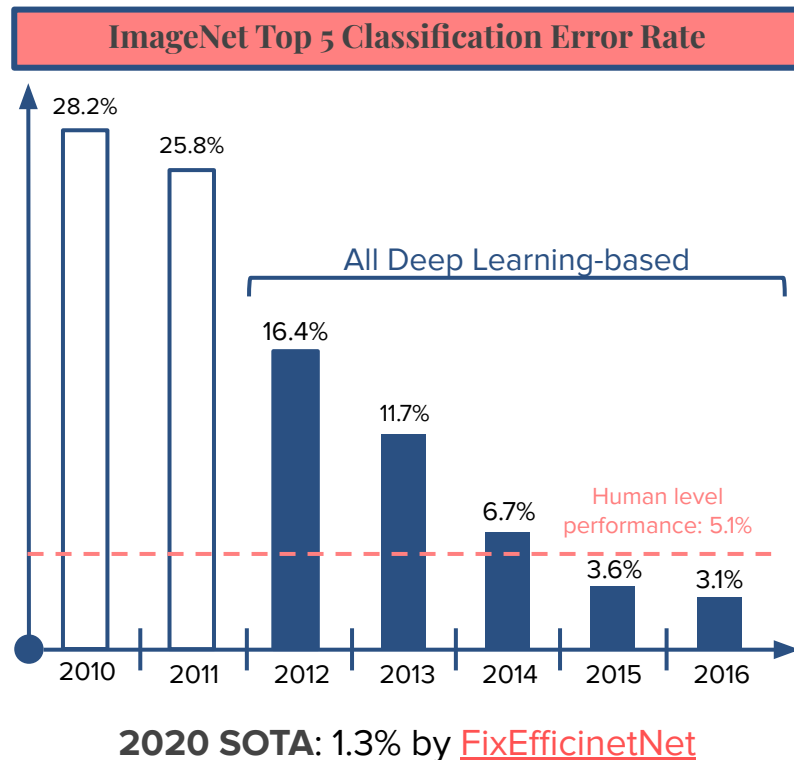


Computer Vision Tasks



Deep Learning for Image Classification

- Last time, we saw how well the Inception Networks perform on ImageNet and how they can learn interesting image features.
- But Inception V3's great result on ImageNet (5.3% Top-5 error) still pales compared to the recent **State-of-The-Art (SOTA)** for that task.
- In fact, every year (now every few months!) we see the next SOTA deep learning model **dethrone** the previous model.
- Furthermore, the Human Performance on it was long outmatched.
- But what do these results really mean?



Adversarial Examples

- In some ways, however it **doesn't mean** that deep learning achieved super-human recognition capacity.
- One way to see this is via **Adversarial Examples**. Consider the following classifications made by GoogLeNet trained on ImageNet:



"panda"
(57.7% confidence)

+ .007 ×



"nematode"
(8.2% confidence)

=



"gibbon"
(99.3% confidence)

- Despite making the right classification for the original image, it gives a very wrong result (with certainty) on a very similar image!

Natural Adversarial Examples

- The last image is adversarial because, despite being *seemingly easy* for a good network to classify well, that network makes a crude mistake.
- We can distinguish two types of adversarial examples: **natural** and **synthetic**.
- A natural adversarial example is a natural, organic image which is tough for the model to comprehend.
- The ImageNet-A dataset was created to be a set of natural images, **easily classified by humans**, that ResNet50 trained on ImageNet (Top-5: 7.8%) classifies very poorly.

Network Predictions Using ResNet-50 on Images from ImageNet-A



Class: Dragonfly
Prediction: Manhole
Cover



Class: Bullfrog
Prediction: Fox
Squirrel



Class: Butterfly
Prediction: Washing
Machine



Class: Jay
Prediction: Jeep

Natural Adversarial Examples

- In-fact the ResNet-50 (the SOTA method for for some years) pre-trained model obtains an **error of 97%** on ImageNet-A!
- The same ImageNet-A's paper also show that this poor classification result is a product of the network using **wrong image cues** when classifying images:

Shape cue



Class: Candle
Prediction:
Jack-o'lantern

Class: Lycaenidae
Prediction: Broom

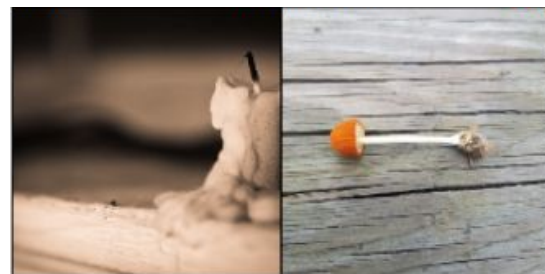
Color cue



Class: Drangonfly
Prediction: Skunk

Class: Drangonfly
Prediction: Banana

Background cue

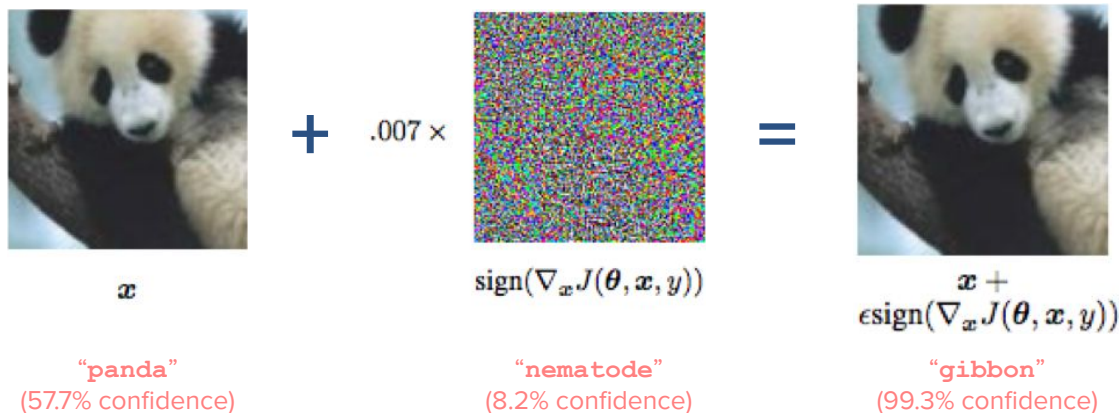


Class: Candle
Prediction: Nail

Class: Mushroom
Prediction: Nail

Synthetic Adversarial Examples

- Besides these naturally occurring adversarial examples, one can also **synthetically** create them.
- Here we artificially **induce some noise** in an image such that it still remains very similar visually to the original, but the infused noise ends up degrading the classifier accuracy.
- This is the case of our first example, found in this [paper](#):

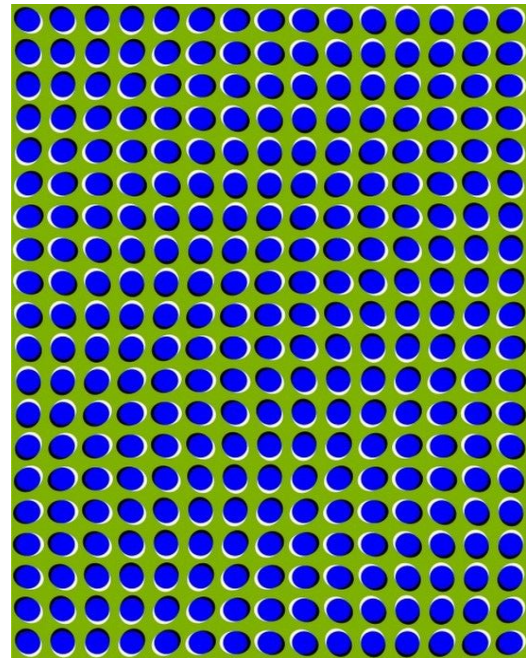


The diagram illustrates the process of creating a synthetic adversarial example. It shows three main components in a sequence: an original image of a panda, a noise pattern, and the resulting adversarial image. The panda image is labeled x and has a confidence of 57.7% for the class "panda". The noise pattern is labeled $\text{sign}(\nabla_x J(\theta, x, y))$ and has a confidence of 8.2% for the class "nematode". The resulting adversarial image is labeled $x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$ and has a confidence of 99.3% for the class "gibbon". The noise pattern is scaled by a factor of 0.007.

$$\begin{matrix} \text{panda image} & + & .007 \times \text{noise pattern} & = & \text{adversarial image} \\ x & & \text{sign}(\nabla_x J(\theta, x, y)) & & x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \\ \text{"panda"} & & \text{"nematode"} & & \text{"gibbon"} \\ (57.7\% \text{ confidence}) & & (8.2\% \text{ confidence}) & & (99.3\% \text{ confidence}) \end{matrix}$$

Synthetic Adversarial Examples

- When generating adversarial examples synthetically, we are creating something that is **analogous to an optical illusion** to humans.
- We explicitly search for the noise pattern that will **break the system**.
- This is done in a strategy similar to what we saw in gradient descent: “How can I change this noise pattern to **maximize the classification error** of the original image?”
- Research also suggests that we **can always** find adversarial examples to any deep learning system due to:
 - NNs are too linear in some regions of the input space ([source](#)),
 - The high dimensionality of its search space ([source](#)),
 - Etc. ([source](#), [source](#)).



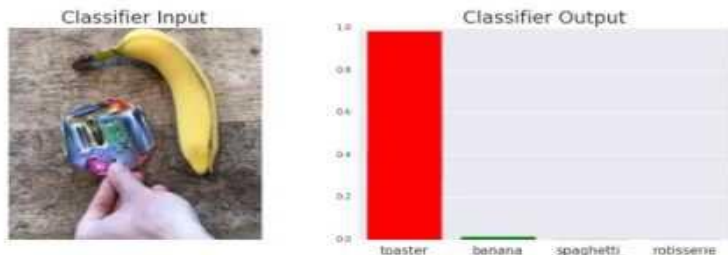
OBS.: The **image** above is not a gif or a video

DL Predictions Are (Mostly) Accurate but Brittle

- The main takeaway is this: deep learning is **very performant**, but also **very brittle**.
- The one simplest solution to improve the performance of one model against adversarial examples is **data augmentation**.
- But [research](#) shows that adding the adversarial data to the training set won't be enough for general tasks (like ImageNet).
- However, **augmentation can work for specific tasks**.

Placing a (weird) sticker on the image can totally change its classification

[Source](#)



DL Predictions Are (Mostly) Accurate but Brittle

- Brittleness of ML is a thing and **adversarial examples can basically always be found**. Should we be worried?
- The quick answer: in some applications, **yes**.

Different perspectives of a Turtle lead to classifying it as a Rifle

[Source](#)



■ classified as turtle ■ classified as rifle
■ classified as other

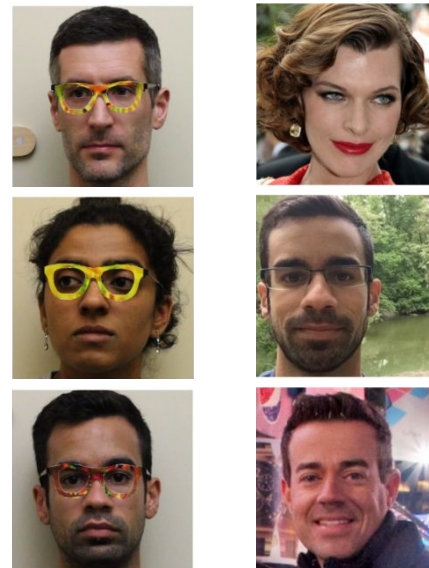
Covering parts of a stop sign lead to wrong classifications

[Source](#)



Mistakes in Face Recog.
because of a glass

[Source](#)



Input Image

Retrieved ID

The issue of Robustness in Deep Learning

- As the world evolve to a more Deep Learning centered world, we find issues to resolve in fields like:
 - **Security:** How can we make software that produces the desired outputs when given the right inputs?
 - **Safety:** How can we ensure that the software is safe for usage, i.e., it does not harm its users (specially in certain applications)?
 - **Alignment:** Need to understand the “failure modes” of Deep Learning, i.e., in which situations/environments the software won't produce the desired outputs with certainty.
- This only elucidate the importance of the study of **robustness** in neural networks, i.e., their ability of **tolerating perturbations** that might affect the system's functions.
- As this issue is critical when applying Deep Learning in many safety-critical and socially-impactful applications, which makes many practitioners **skeptical of DL's future**.
- Research, however, has greatly advance in this field of DL robustness.

Exercises (*In pairs*)

- Which computer vision applications are crucially dependent on robustness? In which ways could you augment their datasets to improve robustness?

What we've seen so far

- So far we noticed a few interesting things about Deep Learning for the task of Image Classification:
 - Deep learning performs very well in classification,
 - The deeper the network, the better the results, but the harder the training,
 - Once the network is trained in some general dataset (like ImageNet), we can use it to solve classification problems in other domains (like cat/dog classification),
 - This process works well because of the good feature learning step deep learning provides.
- Despite the amazing performance of this process, there are two issues it doesn't tackle:
 - Labeled datasets are expensive and time-consuming (ImageNet took 3 years to get labeled). The dataset in itself can be small, with very **few labeled data points**,
 - It may be **very specific** (like in medical imaging) that using features learned from a general dataset may not suffice.
- For these reasons, we cover the task of **Self-supervised learning** today.

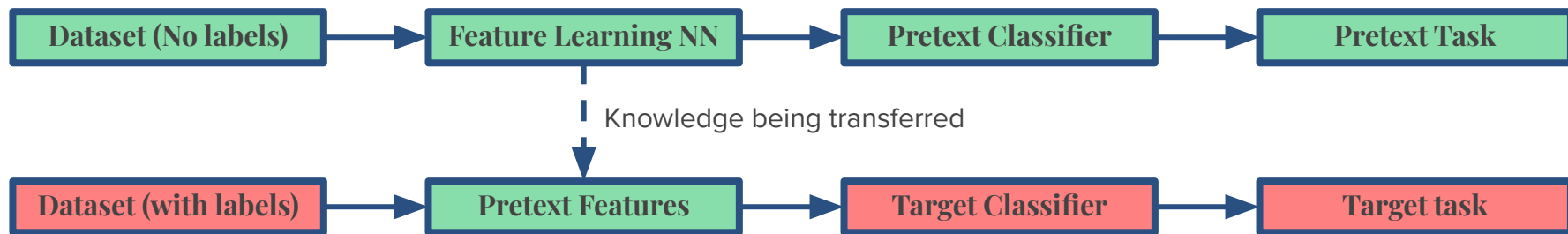
Self-supervision

- The learning in deep learning is based on **Supervised Learning** (SL), where data and labels are available.
- Another way to do learning is via **Unsupervised Learning** (UL), when we only have datapoints (tasks like data clustering and dimensionality reduction).
- One possible middle way between SL and UL is called **Self-Supervised Learning** (SSL), where **the data provides the labels for supervision**.
- SSL is also linked to [how infants learn](#) about the world, hence another reason to do research on it.
- The general strategy for SSL is pre-train the network with a task, called **pretext task**, created with only the datapoints.



Pretext and downstream task

- The aim of the pretext task is to guide the model to learn **intermediate representations** of data, i.e., to do feature learning.
- This is useful in understanding the underlying structural meaning of the data, which will be beneficial for the practical **downstream** (or **target**) tasks.
- The downstream task uses the transfer process of the pretext model to a specific task.



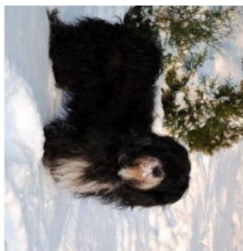
- Many ideas have been proposed by researchers for different image-based tasks to train using the SSL method.

Rotation Classification task

- A simple pretext task for vision problems is **rotation classification**, proposed in 2018.
- Here, the dataset images are rotated by random multiples of 90 degrees (e.g., 0° , 90° , 180° , or 270°) and the network is tasked at detecting the rotation (out of 4 possible).



90°



270°



180°



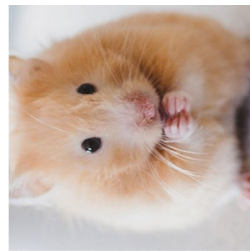
0°



270°



270°

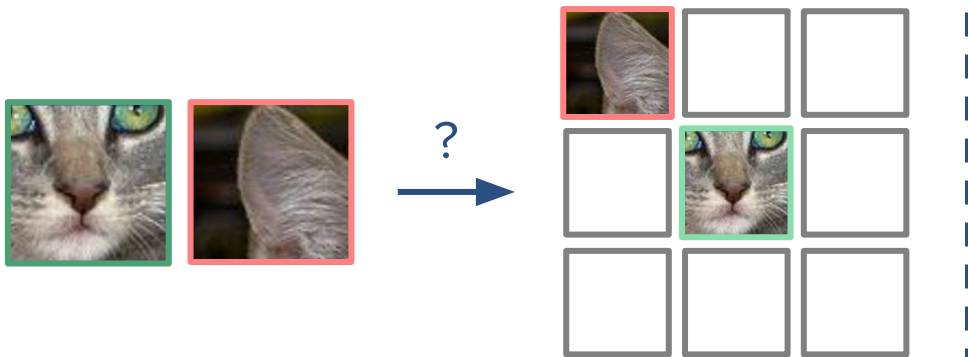


90°

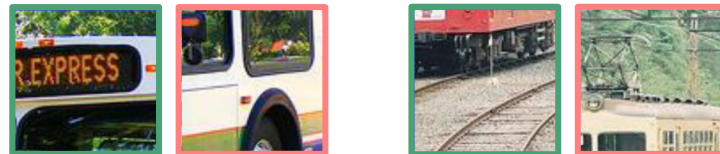
- The authors showed that adding this pretrain step **improved their target classification** step and **also took account for the rotation data augmentation**.
- Furthermore, the pretext task itself is useful in some settings (like detecting if a cell phone is upside down)

Patch Localization Task

- In the Patch Localization task, proposed in 2015, the goal is to **localize an image patch based on another patch**.
- This involves randomly sampling a patch (green border) and then one of eight possible neighbors (red border) and have the network predict its relative position (1 out of 8).



Try it yourself: where are the red ones placed according to the green ones?



- According to the authors, this pretext task would help the network learn **spatial context information** more efficiently.

SimCLR Task

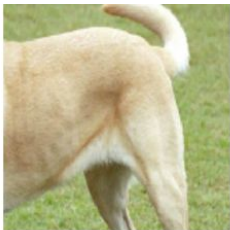
- Another example of pretext task is called SimCLR (**S**imple Framework for **C**ontrastive **L**earning of Visual **R**epresentations), [published](#) in 2020.
- It uses the concept of **Contrastive Learning**, that relies on comparing pairs of dataset images.
- The idea is simple: for each image from the dataset, create a set of augmentations for it.
- Then, train a CNN (ResNet in their case), followed by an MLP, that **maximizes the similarity** between pairs of augmentations from the same image and minimizes it for different images.
- After training, use only the CNN as your feature representer for transfer learning.

SimCLR Task

- The augmentations used in the were cropping, resizing, rotation, noise addition, etc.



Original



Crop + Resize



Crop + Resize + Flip



Distort



Rotate



Cutout



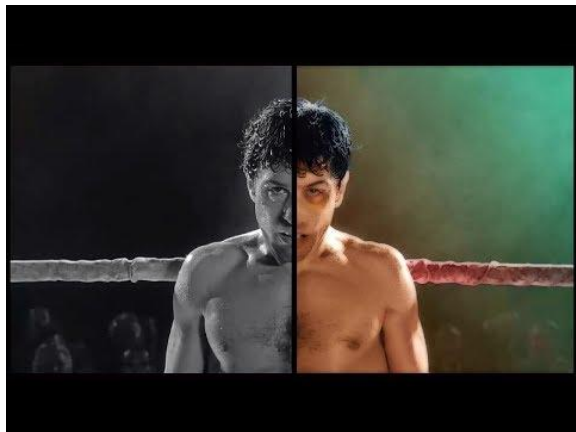
Noise

- The rationale behind these simple transformations of individual images is
 - They wanted to encourage "consistent" representation of the same image under various transformations,
 - Since the pre-training data lacks labels, we can't know a priori which image contains which object class,
 - The authors found that these simple transformations suffice for the neural net to learn good representations.

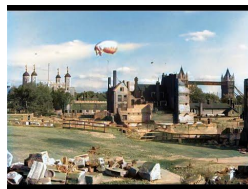
Self-supervised learning beyond Classification

- Summary:
 - Pretext tasks focus on “visual common sense”, e.g., predict rotations, spatial context, etc.
 - The models are forced to learn good visual features in order to solve the pretext tasks.
 - We (usually) don’t care about the performance of these pretext tasks, but rather how useful the learned features are for downstream tasks
- Self-supervised learning has being applied to many other computer vision tasks besides classifications, for example:
 - **Image Inpainting:** fill in missing parts of an image.
 - **Image Semantic Clustering:** group images that are similar in content together in different clusters.
 - **Image Coloring:** turning a grayscale image into an RGB one,
 - **Video Coloring:** same as image coloring but for videos.
- Starting from next class, we’ll study other Computer Vision tasks beyond Image Classification.

Video: *Video Automatic Colorization*



AI Colorized footages of old cities



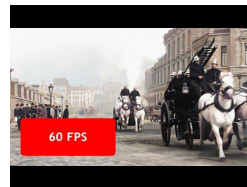
London



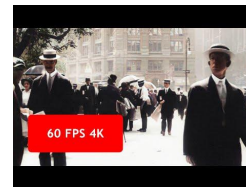
Beijing



Tokyo



Paris



New York