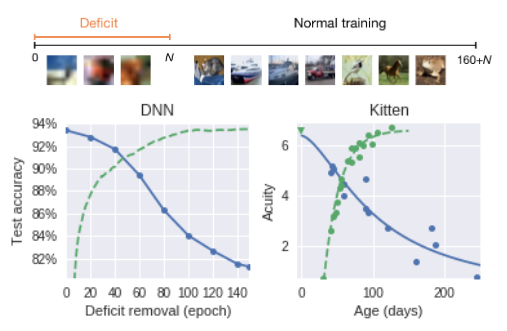
## Experiments from [Achille et al](https://openreview.net/pdf?id=BkeStsCcKQ).

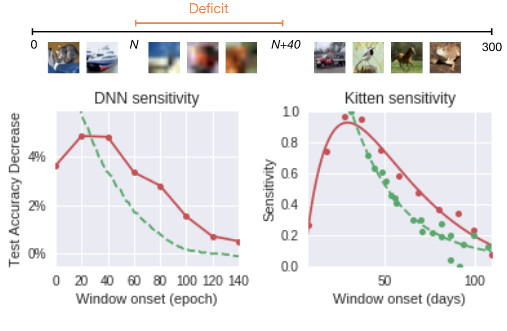
El dataset son las imagenes CIFAR-10

* Experiment 1: To simulate the effect of cataracts, for the first t0 epochs the images in the dataset are [downsampled](https://stackoverflow.com/questions/18666014/downsample-array-in-python) to 8 × 8 and then upsampled back to 32 × 32 using [bilinear interpolation](https://stackoverflow.com/questions/8661537/how-to-perform-bilinear-interpolation-in-python), in practice blurring the image and destroying small-scale details.

After that, the training continues for 160 more epochs, giving the network time to converge and ensuring it is exposed to the same number of uncorrupted images as in the control (t0 = 0) experiment.

Outcome of that: We can readily observe the existence of a critical period: If the blur is not removed within the first 40-60 epochs, the final performance is severely decreased when compared to the baseline (up to a threefold increase in error).

* Experiment 2: We can measure more accurately the sensitivity to a blur deficit during learning by introducing the deficit in a short window of constant length (40 epochs), starting at different epochs, and then measure the decrease in the DNN’s final performance compared to the baseline (Figure 1).

Outcome of that: We observe that the sensitivity to the deficit peaks in the central part of the early rapid learning phase (at around 30 epochs), while introducing the deficit later produces little or no effect.

* Experiment 3 + outcome: We observe that DNNs are also largely unaffected by **high-level deficits** – such as ***vertical flipping of the image, or random permutation of the output labels***: After deficit correction, the network quickly recovers its baseline performance
* Experiment 4 + outcome: We apply to the network a **more drastic deficit**, where ***each image is replaced by white noise***. Figure 2 shows hows this extreme deficit exhibits a remarkably less severe effect than the one obtained by only blurring images: Training the network with white noise does not provide any information on the natural images, and results in milder effects than those caused by a deficit (e.g., image blur), which instead conveys some information, but leads the network to (incorrectly) learn that no fine structure is present in the images.

## Our experiments

* Ariquitectura desde aca <https://github.com/StefOe/all-conv-pytorch/tree/master> y desde ese forum (equivalente) <https://discuss.pytorch.org/t/pytorch-net-from-striving-for-simplicity-the-all-convolutional-net/19297>
* Primero imitar lo de Achille et al. [parece que funciona – blurring.ipnyb]
* Pasar imagenes a Fourier y cortar frecuencias
* Pasar imagenes a Fourier y modular (por ejemplo area bajo la curva constante) para quitar de forma pesada

## Reu 5 (09 Nov 2023):

* Idea Mircea: en vez de tomar imágenes, tomar señales 1D. ¿Cuáles son los beneficios de utilizar imágenes? Recordemos que las imágenes tienen info redundante.
* Es posible que CIFAR10, al tener resolución 32x32 no funcione tan bien la descomposición en frecuencias, por su tamaño muy pequeño. Explorar si buscamos un dataset más grande. Probar igualmente en CIFAR y ver que pasa.
* Pasar imágenes a blanco & negro, para facilitar Fourier.
* Sobre imágenes: Christ utilizaba T. Fourier 2D para aislar frecuencias altas, modificaba las frecuencias de tal manera que todas tengan la misma importancia, respetando área que el bajo la curva se conserve. Es importante considerar que se conserva la fase, por esto no se “destruye” la imagen.
* Dado que tenemos acceso a patagón. ¿Qué tanto más grandes pueden ser nuestras arquitecturas, datasets, etc?
* Investigar que se dice en los foros sobre las frecuencias altas en imágenes pequeñas. También ver antes del miercoles si se deben llegar los datos de forma fisica o solo online.
* Para este finde, revisar patagon y entender los tutoriales.

## Reu 6 (7 Dic 2023)

Tareas:

Tomar (imágenes - blur). De esta manera obtenemos las frecuencias más altas. Luego replicar el experimento de achille y ver cómo afecta.

Utilizar difference of gaussians con high sigma = 4. Probar distintos sigmas.

En resumen, ahora que ya tenemos las llaves de patagon, instalar los contenedores correspondientes para correr torch (tutorial en la página de patagon). Correr experimentos similares al de achille, pero en vez de utilizar el blurr, el cual remueve frecuencias altas, hacer imagen - blur, que remueve frecuencias bajas.

Explicación de difference of gaussians

[Molecular Expressions Microscopy Primer: Digital Image Processing - Difference of Gaussians Edge Enhancement Algorithm - Interactive Tutorial (fsu.edu)](https://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/diffgaussians/)

## Reu 7 (10 Enero 2023)

Partir blur desde época 5

Ver cómo está aprendiendo a reconocer las HF images (testear HF)

Tercera línea de la precisión de altas frecuencias (Lo hice para blur primero)

Añadir batch normailization a la AllConvNet