MÓDULO 3

Estratégias de Treinamento de CNNs

ESPECIALIZAÇÃO
INTELIGÊNCIA ARTIFICIAL
& CIÊNCIA DE DADOS

SEAD UFES | Superintendência de Educação a Distância

Bruno Légora Souza da Silva
Professor do Departamento de Informática/UFES

ÍNDICE

1. Problemas no Treinamento de CNNs
2. Estratégias
   - 2.1. Adequação de Resolução
   - 2.2. Aumento de Dados
   - 2.3. Uso de *Loaders*
   - 2.4. Transferência de Aprendizado
3. Laboratório 8

1. Problemas no Treinamento de CNNs

Problemas no Treinamento de CNNs

- Em geral, o treinamento de CNNs (e de outras arquiteturas) possuem alguns problemas.
  - Um deles já foi adiantado no Lab 7 - o problema de resoluções diferentes!

Problemas no Treinamento de CNNs

- Outro grande problema é relacionado ao grande tamanho das arquiteturas:
  - Muitos dados são necessários para treinar tais redes
    - IMAGENET tinha 1M imagens...

Problemas no Treinamento de CNNs

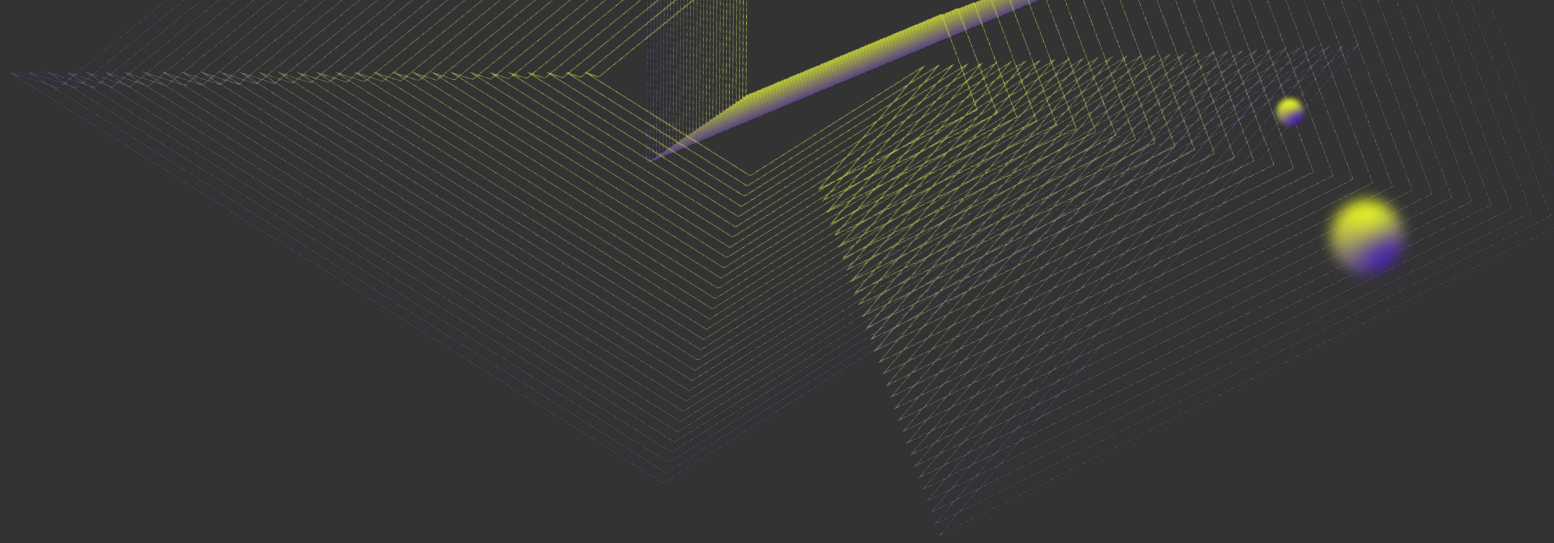
- Em muitos casos, não temos tantas imagens marcadas/classificadas, como é o caso da Imagenet
  - Nem sempre o treinamento fica bom...
  - Mas podemos tentar reduzir esse problema!

Problemas no Treinamento de CNNs

- E, além disso, com grandes quantidades de imagens, geralmente precisamos de uma grande infraestrutura de hardware
  - Ou aplicamos técnicas para carregá-las aos poucos

Problemas no Treinamento de CNNs

- Por último, a tarefa de treinamento é “pesada”.
- É possível aproveitar redes já prontas em outras aplicações, apenas “ajustando” elas para o novo problema!



2 Estratégias utilizadas

Soluções para os problemas no Treinamento de CNNs

- Veremos 4 (aqui e nos Labs 8-11):
  - Adequação de Resolução
  - Aumento de Dados
  - Uso de *Loaders*
  - Transferência de Aprendizado

2.1 Adequação de Resolução

Adequação de Resolução

- No Lab. 7, vimos que a base de dados possuía imagens de tamanhos distintos
  - A princípio, não é um problema pois ao contrário das outras arquiteturas vistas na Aula 7, a U-Net processa todas elas!

Adequação de Resolução

- Porém, treiná-las pode ser um problema:
  - Precisamos passar as imagens em “batches”, que no pior caso tem tamanho 1 (como fizemos no exemplo)

Adequação de Resolução

- Porém, treiná-las pode ser um problema:
  - Todo método de treinamento é uma descida de gradiente
  - O gradiente calculado com uma única imagem é potencialmente instável

Adequação de Resolução

- Porém, treiná-las pode ser um problema:
  - Isso faz com o que o treinamento fique “perdido” e possivelmente não chegue à convergência.

Adequação de Resolução

- Para solucionar esse problema, podemos adotar duas estratégias:
  1. Escolher uma resolução fixa e aplicar a operação de redimensionamento em todas as imagens

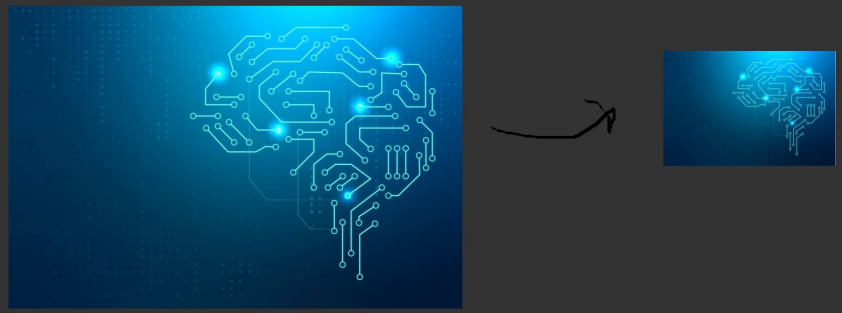
Adequação de Resolução

- Para solucionar esse problema, podemos adotar duas estratégias:
  1. Ex: Redimensionar todas as imagens (tanto entrada como saída) para 560x560, no caso da U-net

Adequação de Resolução

- Para solucionar esse problema, podemos adotar duas estratégias:
  1. Esse procedimento pode inserir ruído nas imagens, mas o treinamento tende a ficar melhor

Adequação de Resolução



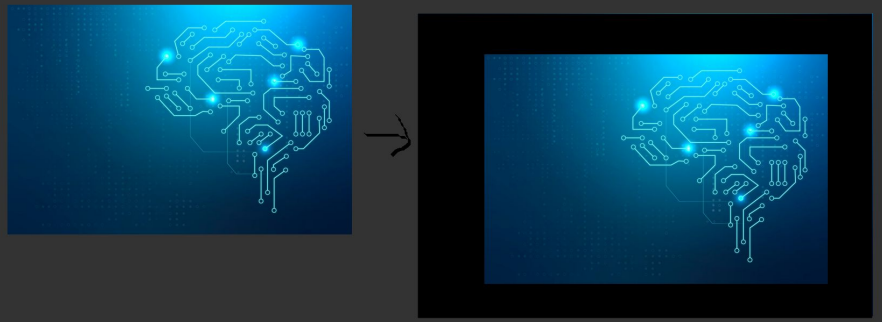
Adequação de Resolução

- Para solucionar esse problema, podemos adotar duas estratégias:
  2. Escolher a maior resolução do conjunto e criarmos bordas (padding) em todas as imagens antes de treinar/usar a rede.

Adequação de Resolução

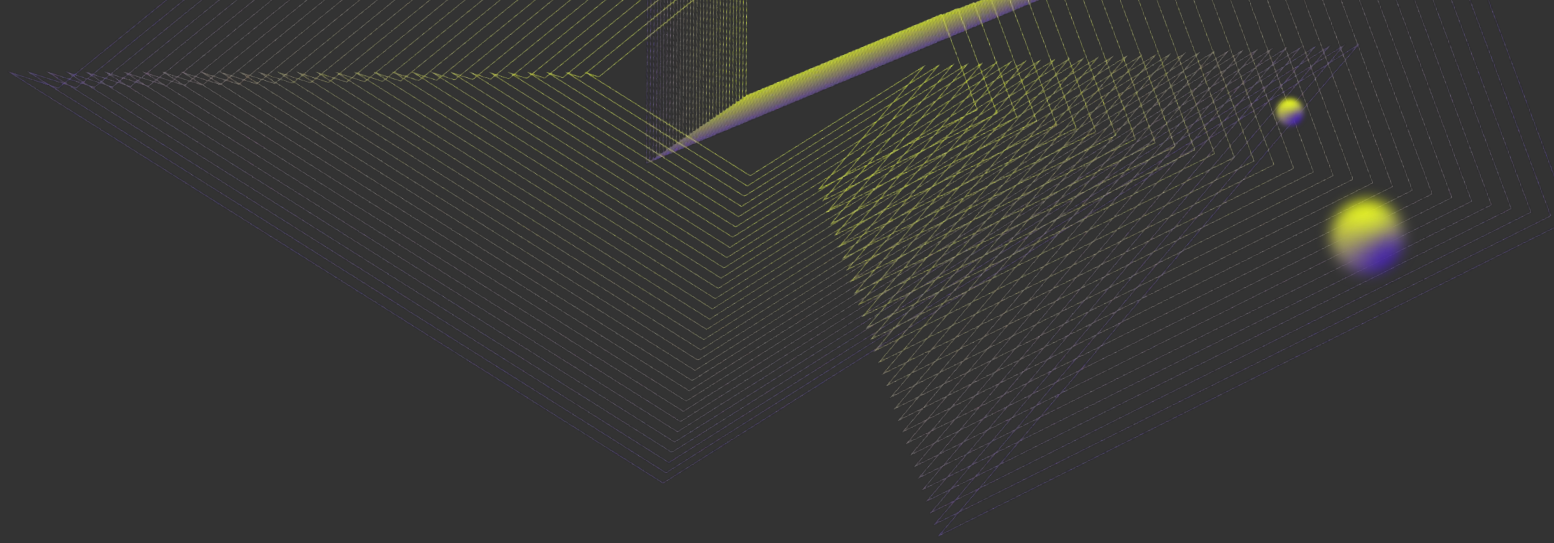
- Para solucionar esse problema, podemos adotar duas estratégias:
  2. Após passar pela rede, a imagem é cortada para a resolução original dela, caso necessário

Adequação de Resolução



Adequação de Resolução

- Ambas as formas são válidas (é possível até escolher um meio termo entre elas) e “inserem ruído” na base de dados, mas o processo de treinamento tende a ficar mais otimizado e a rede consegue aprender mais!



2.2 Aumento de Dados

Aumento de Dados

- A técnica de Aumento de Dados (do inglês *Data Augmentation*) consiste em criar imagens sintéticas com base nas imagens originais.

Aumento de Dados

- Essas imagens sintéticas são criadas após utilizarmos, por exemplo, as transformações afim que vimos no início da disciplina!

Aumento de Dados

- Partimos do pressuposto que uma imagem, ao passar por uma leve transformação (como rotação ou deslocamento) não altera o seu conteúdo (cuidado!)

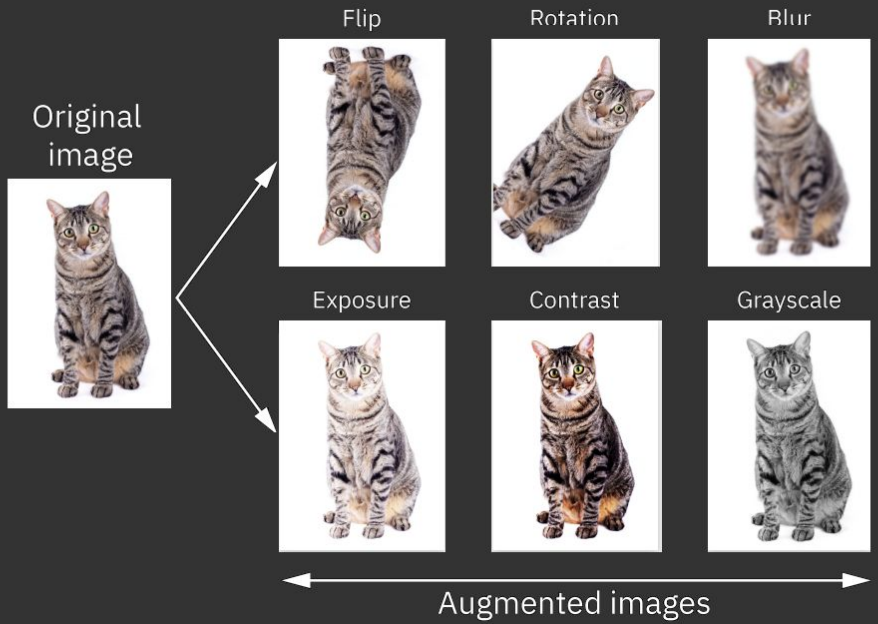
Aumento de Dados

- Para problemas de classificação, rotacionar uma imagem em 15º não irá alterar a sua classe;
- Para problemas de segmentação, rotacionar uma imagem em 15º irá alterar a segmentação em 15º!
  - Mas, muito cuidado;

Aumento de Dados

- Exemplos de transformações:
  - Rotação
  - Translação
  - Cor
  - “Flip”
  - “Borramento”

Aumento de Dados



Aumento de Dados

- Cuidado! Isso pode não ser uma boa ideia em determinadas aplicações!
- Exemplo:

Aumento de Dados

Aumento de Dados

- No caso das placas de trânsito, rotacionar uma placa pode fazer a rede confundi-la com outra diferente!
- Neste caso, talvez seja melhor não fazer rotações e nem operações com cor...

2.3 Uso de *Loaders*

Uso de Loaders

- Grandes bases de dados requerem hardware capaz de processá-las.
  - Basta comprar mais memória...
  - Ou usar *Loaders* no seu hardware mais modesto!

Uso de Loaders

- *Loaders* geralmente são classes fornecidas por bibliotecas de IA que funcionam como “wrappers” de uma base de dados;

Uso de Loaders

- Em geral fornecem métodos que facilitam o carregamento da base de dados em “pedaços”
- Muito útil para gerenciar grandes bases em computadores com “pouca” memória!

Uso de Loaders

- Em geral fornecem métodos que facilitam o carregamento da base de dados em “pedaços” (*batches*)
- Muito úteis para gerenciar grandes bases em computadores com “pouca” memória!

Uso de Loaders

- Ao invés de ter toda a base carregada na memória, carregamos ela sob demanda*:*
  - Ex: Salva os caminhos dos arquivos e carrega os arquivos na memória apenas quando necessário

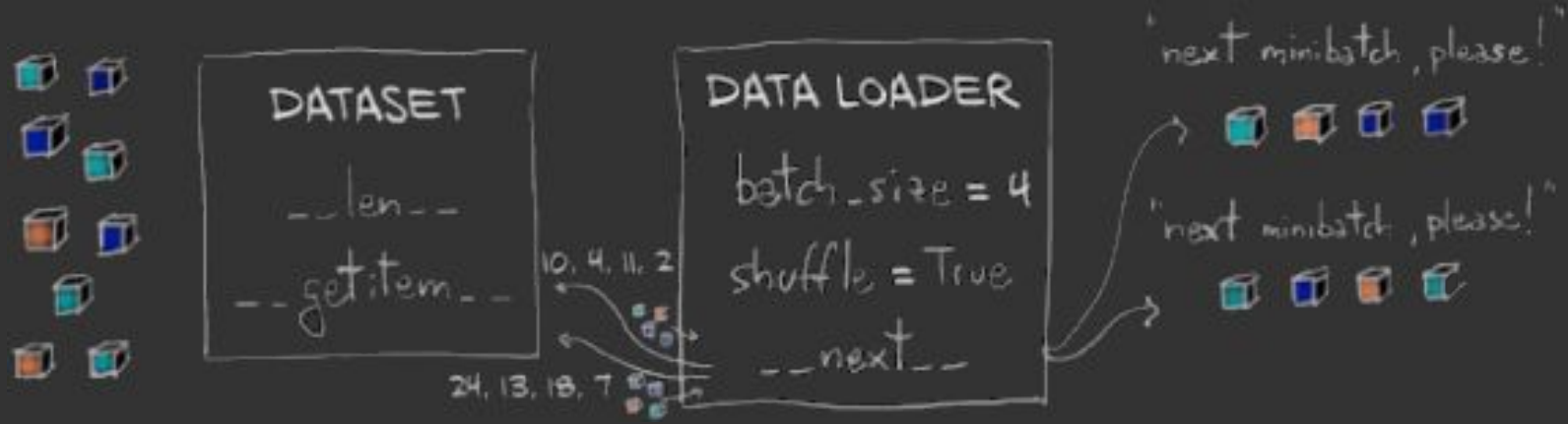
Uso de Loaders

- Também podemos fazer outras etapas dentro do próprio *loader:*
  - Ajusta a resolução;
  - Realiza o aumento de dados;
  - Outras operações (ex: *shuffle*)

Uso de Loaders

- O loader então retorna o *batch* pré-processado, que é usado para treinar a rede e depois “descartado” da memória, dando lugar ao próximo *batch;*

Uso de Loaders



Uso de Loaders

- No Lab. 7, usamos uma base de dados que não era possível de ser carregada na memória do Google Colab.
  - Usamos apenas 10% dela.
- No Lab. 10, iremos usar loaders para treinar a U-net com a base de dados completa;

2.4 Transferência de Aprendizado

Transferência de Aprendizado

- Por fim, veremos a técnica de Transferência de Aprendizado (do inglês, *Transfer Learning*).

Transferência de Aprendizado

- O treinamento de rede neural geralmente consiste em:
  - Definir uma arquitetura;
  - Inicializar a arquitetura com pesos aleatórios;
  - Treinar o modelo com os dados;

Transferência de Aprendizado

- O fato de inicializar os pesos aleatoriamente indica que a rede neural deve aprender “do zero”, pois ela não tem conhecimento algum sobre nada;
  - Como se ela “chutasse” sempre

Transferência de Aprendizado

- Além disso, será que cada tarefa precisa utilizar uma arquitetura diferente? Ou podemos usar a U-net para Segmentação de Imagens da Imagenet e também para imagens médicas, por exemplo?

Transferência de Aprendizado

- Não é necessário criar uma arquitetura para cada tarefa;
- Além disso, modelos treinados estão disponibilizados na internet;
- Por que não usá-los para outras tarefas?

Transferência de Aprendizado

- Assim, podemos utilizar a estratégia chamada de *Transfer Learning*, que é basicamente um *fine-tuning* de uma rede neural já treinada, mas para outro problema;

Transferência de Aprendizado

- Essa técnica parte do pressuposto que, por exemplo, a U-net usada na base Imagenet possui algum conhecimento que possa ser aproveitado numa base de dados de imagens médicas!

Transferência de Aprendizado

- Ao invés de iniciar a rede neural com pesos aleatórios, fazemos:
  - Usamos uma arquitetura já existente;
  - Carregamos os pesos que podem ser baixados de algum repositório da internet!
  - E treinamos na nossa base de dados!

Transferência de Aprendizado

- A tendência é que o *fine-tuning* seja mais rápido do que o treinamento “do zero”. Veremos isso em mais detalhes no Laboratório 11!

Transferência de Aprendizado

- Para o caso da U-net, cujo tamanho da imagem de saída é proporcional ao tamanho da imagem de entrada, não há “problemas” em usar a rede diretamente.
- Podemos até usá-la para outro tipo de tarefa.

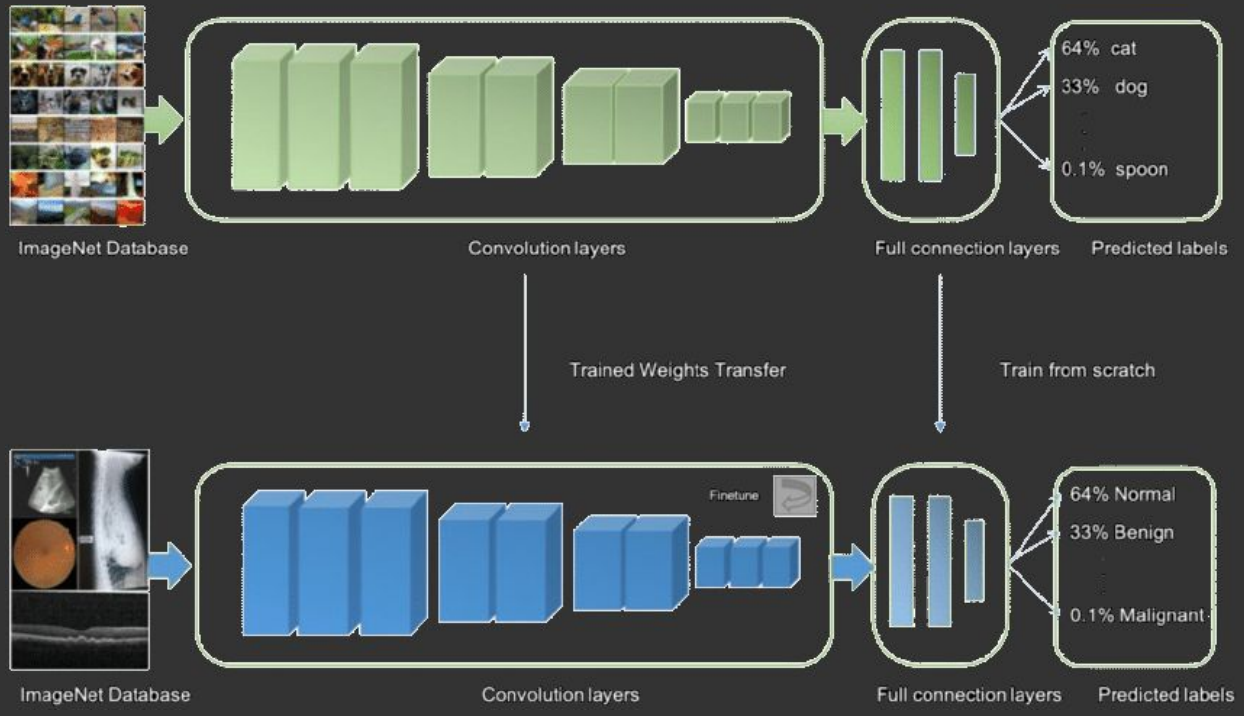
Transferência de Aprendizado

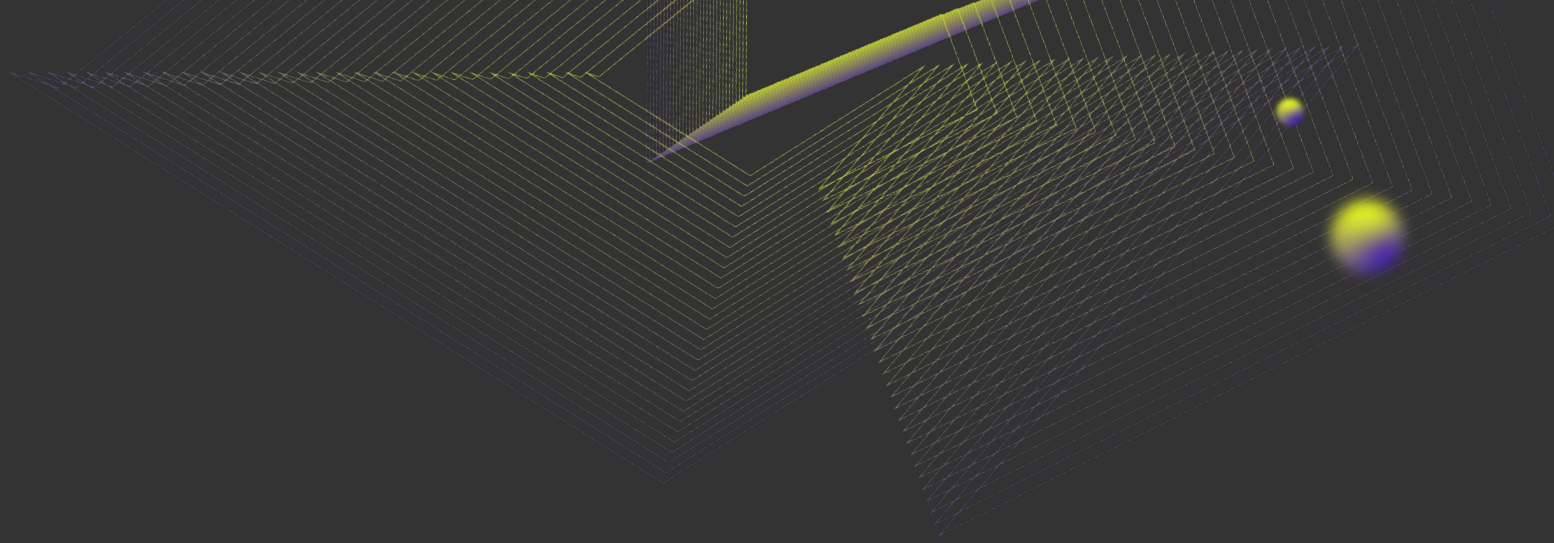
- Para o caso de redes que foram treinadas para classificar imagens em 1000 classes (alexnet, inception, resnet, etc), é possível aproveitar parte de seus pesos (por exemplo, substituindo apenas a última camada, que “define” o número de classes de saída);

Transferência de Aprendizado

- Os pesos aproveitados podem (ou não) ser “travados” - não serão alterados pelo processo de treinamento, se for desejado pelo programador.
    - Neste caso, o *fine-tuning* fica mais rápido ainda!

Transferência de Aprendizado





3. Laboratórios 08 a 11

Laboratório 8-11

- No 8º laboratório da disciplina, vocês voltarão ao treinamento da U-net, mas padronizando as resoluções.
- No Moodle!

Laboratório 8-11

- No 9º laboratório da disciplina, vocês voltarão ao treinamento da U-net, mas usando data augmentation;
- No Moodle!

Laboratório 8-11

- No 10º laboratório da disciplina, vocês voltarão ao treinamento da U-net, mas usando *loaders*;
- No Moodle!

Laboratório 8-11

- No 11º laboratório da disciplina, vocês voltarão ao treinamento da U-net, mas fazendo *transfer learning;*
- No Moodle!

Laboratório 8-11

- Nessa semana, além dos laboratórios, temos o quinto e último exercício avaliativo (EA5) da disciplina!

INTELIGÊNCIA ARTIFICIAL & CIÊNCIA DE DADOS

Bruno Légora Souza da Silva
Professor do Departamento de Informática/UFES

bruno.l.silva@ufes.br

SEAD UFES
Superintendência de Educação a Distância