Développeur d'Intelligence Artificielle Appliquée

Cours #12

www.impactia.org

Structure de la formation

- **#1: Introduction**
- **#2: Vision**
- **#3: Vision**
- **#4: Vision**
- **#5: Renforcement**
- **#6: Renforcement**
- **#7: Renforcement**
- **#8: Langage**
- **#9: Langage**
- **#10: Projet**
- **#11: Langage**
- **#12: IA génératif/Projet**
- #13: IA génératif
- #14: IA génératif
- #15: Présentation projet

Semaines dernières : Cours #11

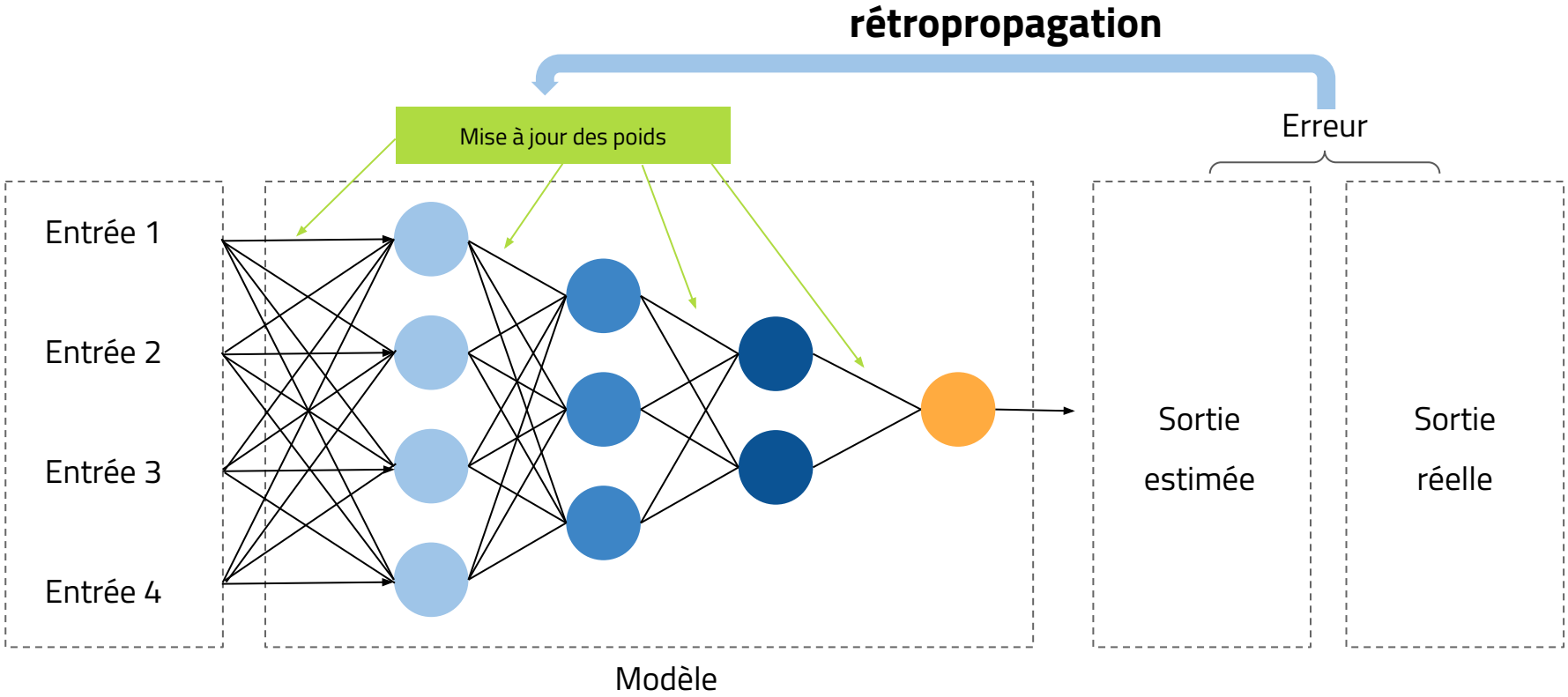
- Génération de texte
  - Avec Python et PyTorch
  - De zéro !

Aujourd’hui : Cours #12

- Pratique
  - …avec Mistral
  - …avec GPT-4
- Théorie
  - Panoramas des modèles de langues actuels
  - Modèles multi-modaux
  - Comment sont entraînés les modèles modernes ?
- Projet
  - Définitions des derniers projets
  - Travail sur les projets

Apprentissage Machine

Réseaux neuronaux



Version facile (mais plus limitée)

```
[ ]  from keras import layers

     model = keras.Sequential(
         [
             layers.Flatten(input_shape=(28, 28)),
             layers.Dense(200, activation="relu"),
             layers.Dense(100, activation="relu"),
             layers.Dense(75, activation="relu"),
             layers.Dense(num_classes, activation="softmax"),
         ]
     )
```


```
[ ]  model.compile(
         loss="categorical_crossentropy",
         optimizer="adam",
         metrics=["accuracy"]
     )
```

```
[ ]  model.fit(
         x_train, y_train,
         batch_size=128,
         epochs=3,
         validation_split=0.1
     )
```

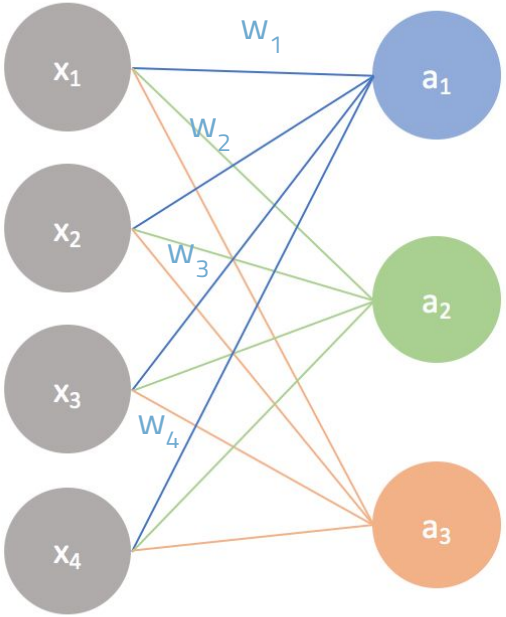
```
[ ]  model.evaluate(x_test, y_test, verbose=0)
```

Version difficile (mais plus de flexibité)

In [795…

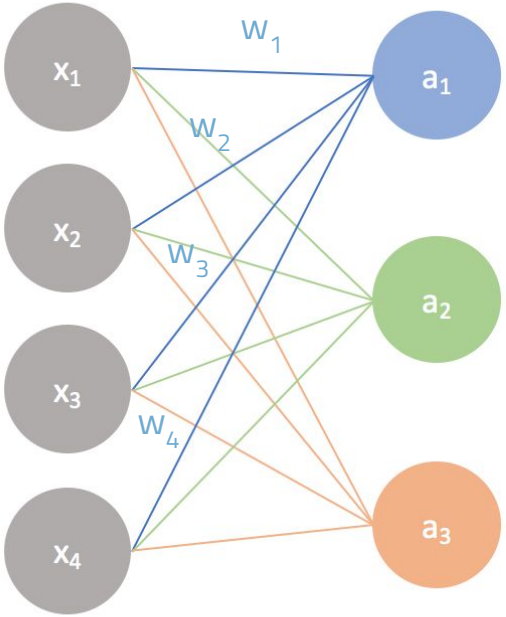
```
for i in range(200000):

  # minibatch construct
  ix = torch.randint(0, Xtr.shape[0], (32,))

  # forward pass
  emb = C[Xtr[ix]] # (32, 3, 10)
  h = torch.tanh(emb.view(-1, 30) @ W1 + b1) # (32, 200)
  logits = h @ W2 + b2 # (32, 27)
  loss = F.cross_entropy(logits, Ytr[ix])
  #print(loss.item())

  # backward pass
  for p in parameters:
    p.grad = None
  loss.backward()

  # update
  #lr = lrs[i]
  lr = 0.1 if i < 100000 else 0.01
  for p in parameters:
    p.data += -lr * p.grad

  # track stats
  #lri.append(lre[i])
  stepi.append(i)
  lossi.append(loss.log10().item())

#print(loss.item())
```

Calculer la sortie d'un réseau de neurones



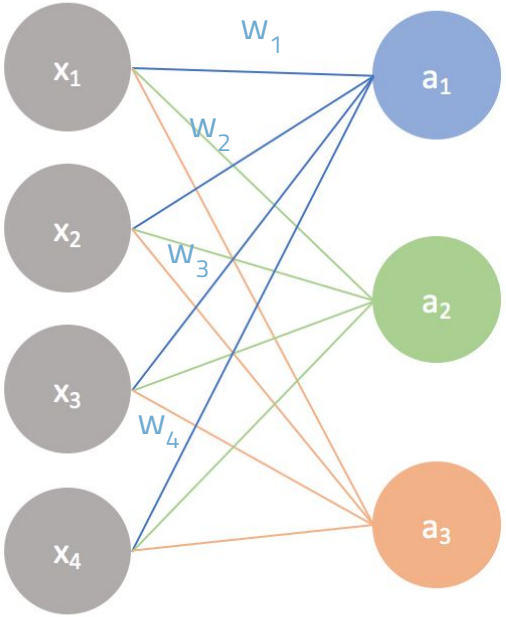
$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Calculer la sortie d'un réseau de neurones



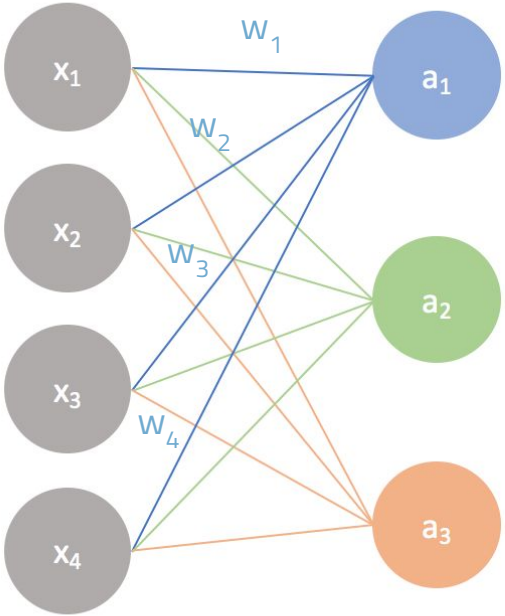
$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \end{bmatrix}$$

Calculer la sortie d'un réseau de neurones



$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Calculer la sortie d'un réseau de neurones

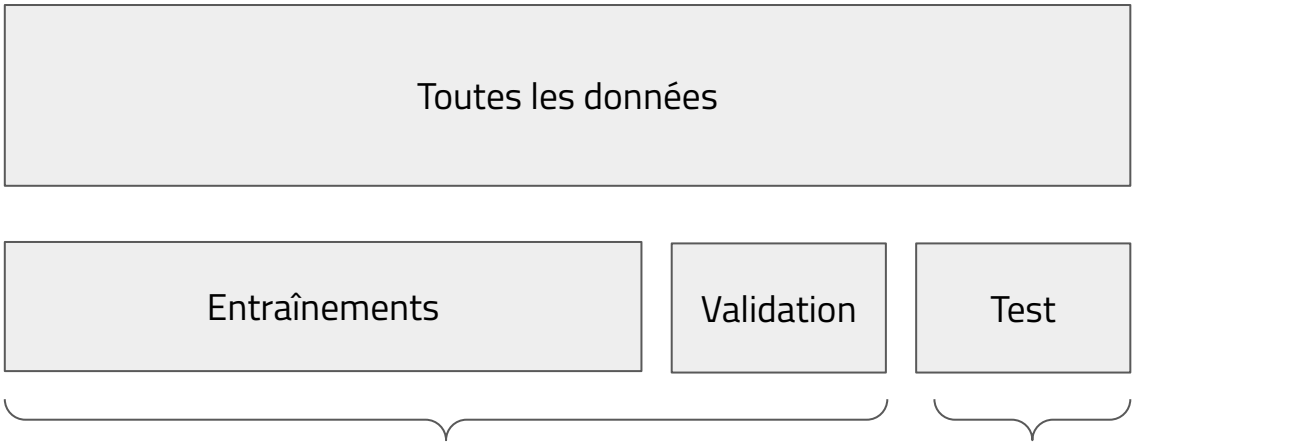


$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

= w @ x

Multiplication de matrices

Jeux de données : “train”, “validate”, “test”



Jeux de données : “train”, “validate”, “test”

```
[ ]  import keras
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[ ]  (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

```
[ ]  print("Dimension de x_train:", x_train.shape)
     print(x_train.shape[0], "images d'entrainement")
     print(x_test.shape[0], "images de test")
```

```
[ ]  x_train = x_train.astype("float32") / 255
     x_test = x_test.astype("float32") / 255
```

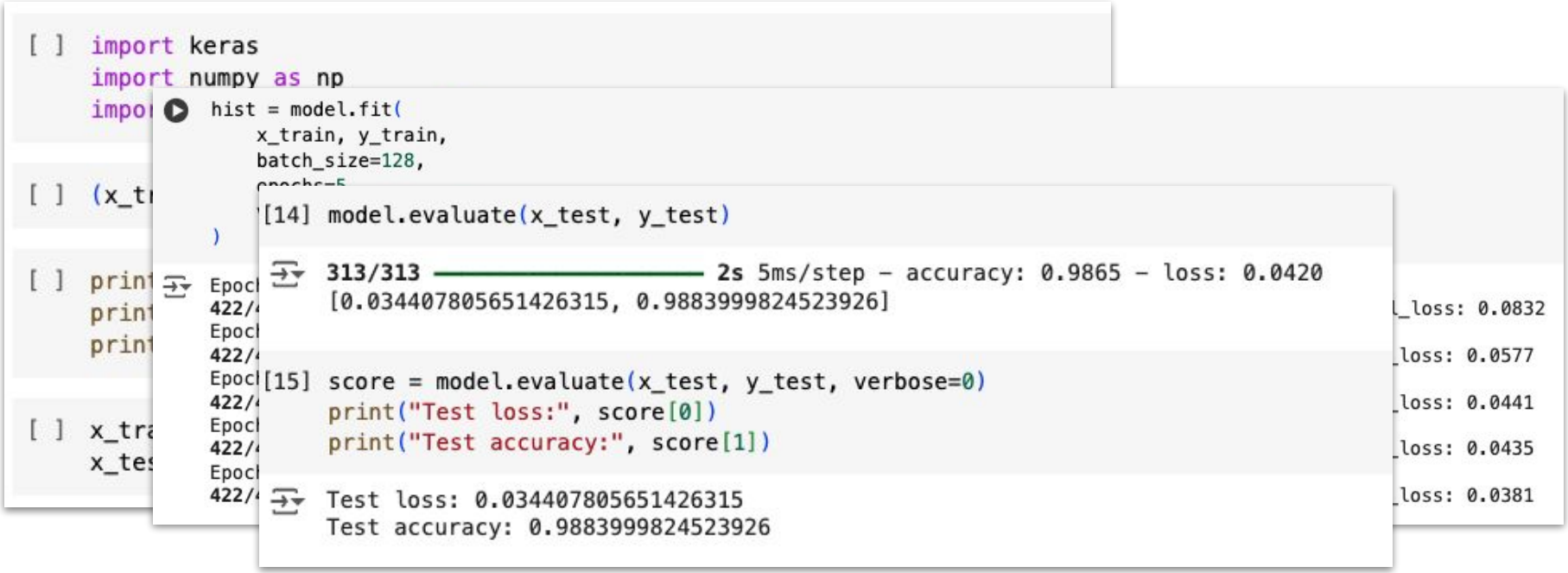
Jeux de données : “train”, “validate”, “test”

```
[ ] import keras
    import numpy as np
    impor

[ ] (x_tr

[ ] print
    print
    print

[ ] x_tra
    x_tes
```

```
hist = model.fit(
    x_train, y_train,
    batch_size=128,
    epochs=5,
    validation_split=0.1
)
```

```
Epoch 1/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 13s 22ms/step - accuracy: 0.7526 - loss: 0.7826 - val_accuracy: 0.9782 - val_loss: 0.0832
Epoch 2/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 9s 20ms/step - accuracy: 0.9638 - loss: 0.1209 - val_accuracy: 0.9840 - val_loss: 0.0577
Epoch 3/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 7s 18ms/step - accuracy: 0.9725 - loss: 0.0910 - val_accuracy: 0.9878 - val_loss: 0.0441
Epoch 4/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - accuracy: 0.9793 - loss: 0.0686 - val_accuracy: 0.9885 - val_loss: 0.0435
Epoch 5/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 5s 12ms/step - accuracy: 0.9792 - loss: 0.0660 - val_accuracy: 0.9897 - val_loss: 0.0381
```

Jeux de données : “train”, “validate”, “test”



Les “batches”

```
[ ]  model.compile(
         loss="categorical_crossentropy",
         optimizer="adam",
         metrics=["accuracy"]
     )
```

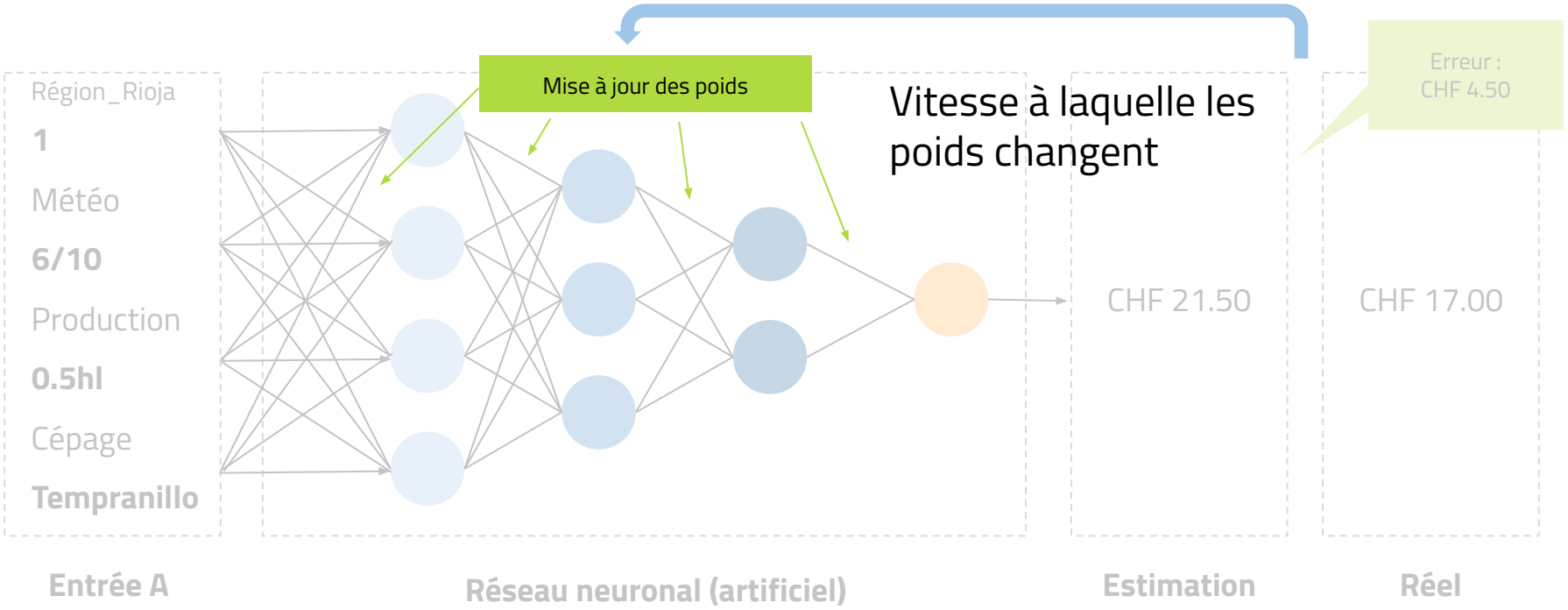
```
[ ]  model.fit(
         x_train, y_train,
         batch_size=128,
         epochs=3,
         validation_split=0.1
     )
```

```
[ ]  model.evaluate(x_test, y_test, verbose=0)
```

```
# Création des agents
agents = {drone.index: RandomAgent(env) for drone in env.drones}
agents[0] = DQNAgent(
    env,
    ConvQNetworkFactory(
        env,
        conv_layers=[
            {'out_channels': 32, 'kernel_size': 3, 'stride': 1, 'padding': 1},
            {'out_channels': 32, 'kernel_size': 3, 'stride': 1, 'padding': 1},
            {'out_channels': 64, 'kernel_size': 3, 'stride': 1, 'padding': 1},
            {'out_channels': 64, 'kernel_size': 3, 'stride': 1, 'padding': 1},
        ],
        dense_layers=[
            1024, 256
        ]),
    gamma=0.95,
    epsilon_start=1,
    epsilon_decay=0.99,
    epsilon_end=0.01,
    memory_size=10000,
    batch_size=64,
    target_update_interval=5
)
trainer = MultiAgentTrainer(env, agents, reset_agents=True, seed=0)
agents[0].qnetwork
```

```
# Entrainement
# on appelle trainer.train plusieurs fois pour changer d'environnement
for run in range(4):
  trainer.train(1000)
```

Learning rate (“taux d’apprentissage”)



Learning rate (“taux d’apprentissage”)

```python
import keras
from keras import layers

def build_model(num_classes):
    inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))

    # On retire les dernières couches avec "include_top=False"
    model = EfficientNetB0(include_top=False, input_tensor=inputs, weights="imagenet")

    # "Glace" the paramètres pré-entrainés
    model.trainable = False

    # On rajoute les couches retirées - ces couches sont entrainables !
    x = layers.GlobalAveragePooling2D(name="avg_pool")(model.output)

    # On peut ajuster le ratio de neurones désactivés pour la couche Dropout
    top_dropout_rate = 0.2
    x = layers.Dropout(top_dropout_rate, name="top_dropout")(x)
    outputs = layers.Dense(num_classes, activation="softmax", name="predictions")(x)

    # Compile le nouveau modèle
    model = keras.Model(inputs, outputs, name="EfficientNet")

    # On augmente le learning rate de 0.001 (valeur par défaut) à 0.01
    optimizer = keras.optimizers.Adam(learning_rate=1e-2)

    model.compile(
        optimizer=optimizer,
        loss="categorical_crossentropy",
        metrics=["accuracy"]
    )
    return model
```

Utilisation du GPU

```
[ ]  from keras import layers

     model = keras.Sequential(
         [
             layers.Flatten(input_shape=(28, 28)),
             layers.Dense(200, activation="relu"),
             layers.Dense(100, activation="relu"),
             layers.Dense(75, activation="relu"),
             layers.Dense(num_classes, activation="softmax"),
         ]
     )
```


```
[ ]  model.compile(
         loss="categorical_crossentropy",
         optimizer="adam",
         metrics=["accuracy"]
     )
```

```
[ ]  model.fit(
         x_train, y_train,
         batch_size=128,
         epochs=3,
         validation_split=0.1
     )
```

```
[ ]  model.evaluate(x_test, y_test, verbose=0)
```

Utilisation du GPU

```
[ ]  from keras import layers

     model = keras.Sequential(
         [
             layers.Flatten(input_shape=(28, 28)),
             layers.Dense(200, activation="relu"),
             layers.Dense(100, activation="relu"),
             layers.Dense(75, activation="relu"),
             layers.Dense(num_classes, activation="softmax"),
         ]
     )
```


Automatique avec Keras !

```
[ ]  model.compile(
         loss="categorical_crossentropy",
         optimizer="adam",
         metrics=["accuracy"]
     )

[ ]  model.fit(
         x_train, y_train,
         batch_size=128,
         epochs=3,
         validation_split=0.1
     )

[ ]  model.evaluate(x_test, y_test, verbose=0)
```

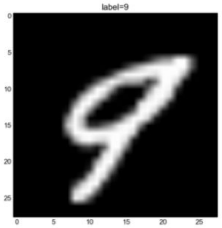
Utilisation du GPU

```
device = 'cpu'
g = torch.Generator(device=device).manual_seed(2147483647)

C = torch.randn((num_chars, 10), generator=g, device=device)
W1 = torch.randn((30, 200), generator=g, device=device)
b1 = torch.randn(200, generator=g, device=device)
W2 = torch.randn((200, num_chars), generator=g, device=device)
b2 = torch.randn(num_chars, generator=g, device=device)
parameters = [C, W1, b1, W2, b2]
```

Avec PyTorch : controle précis du “device”

Encodage “one-hot”

Image :

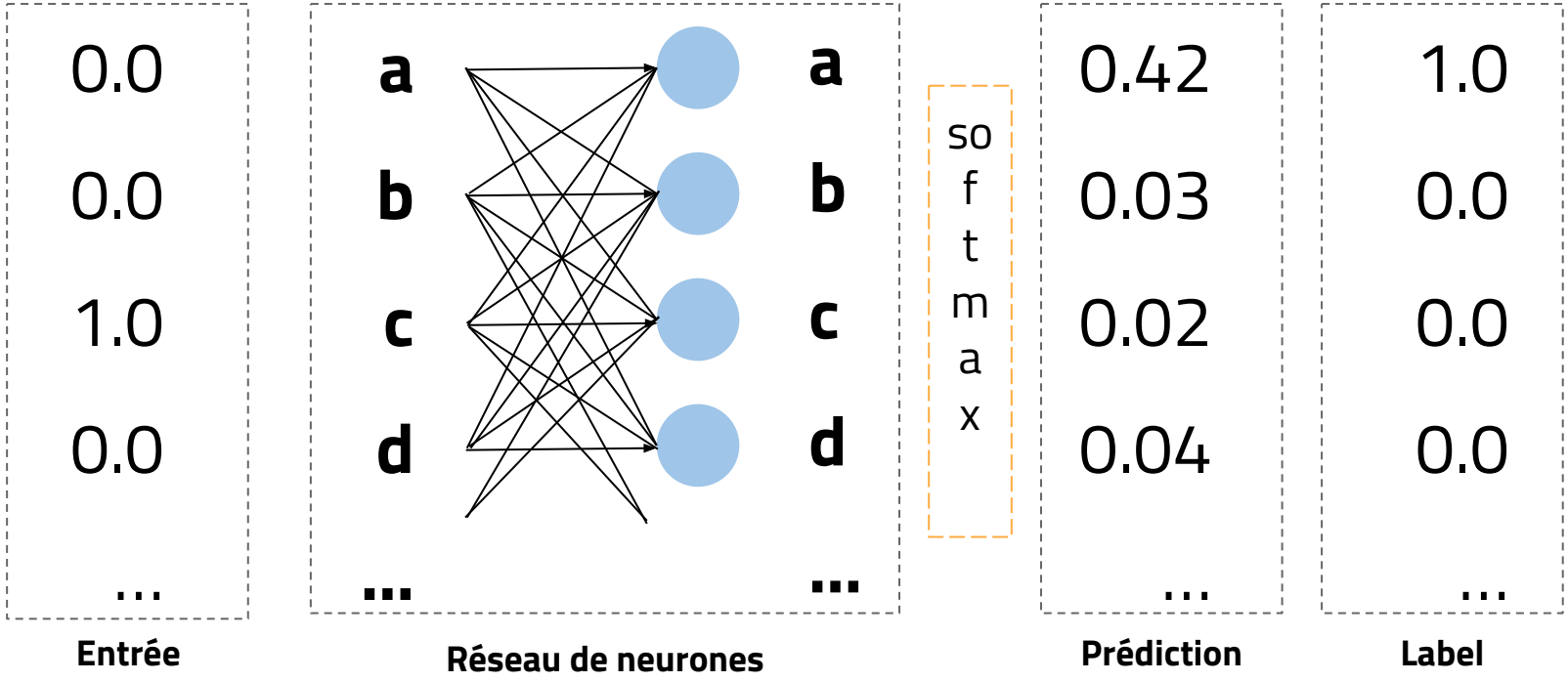


Label : **9**

Label en “one-hot” : `0,0,0,0,0,0,0,0,0,1`

Encodage “one-hot”

Si on a 40 caractères : besoin de 40 entrées



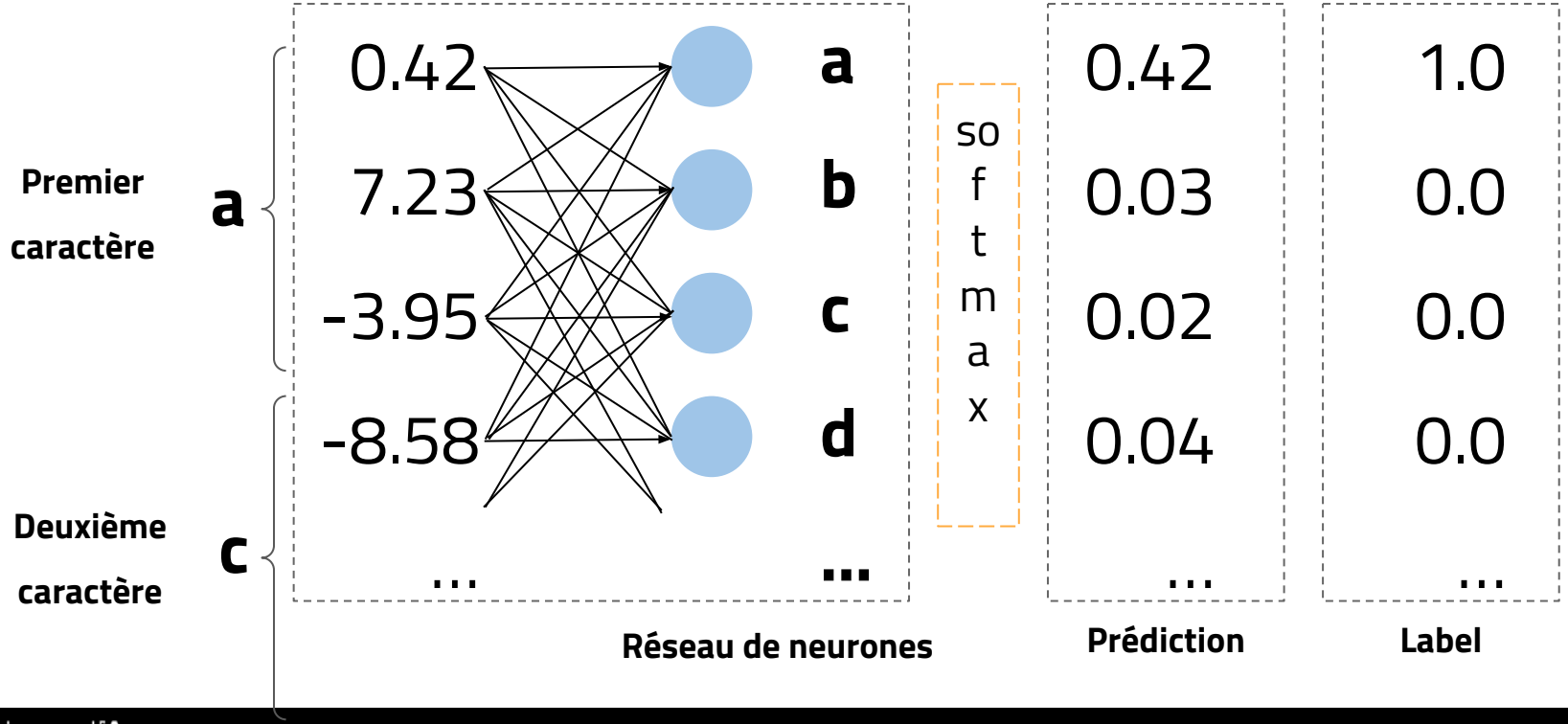
Encodage “one-hot”

Solution : à la place de “one-hot”, on assigne un certains nombre de valeurs à chaque caractère

a	0.42	**b**	2.73	**c**	-8.58	**…**
	7.23		7.72		5.69	
	-3.95		-1.60		-1.76	

Encodage “one-hot”

Si on a 40 caractères : besoin de 40 entrées



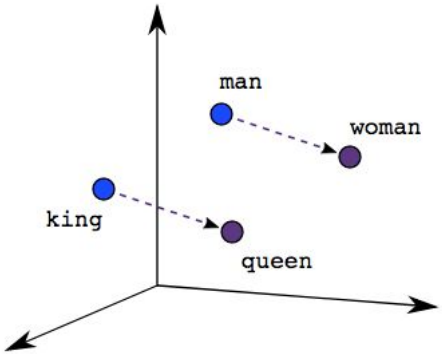
Encodage “one-hot”

Solution : à la place de “one-hot”, on assigne un certains nombre de valeurs à chaque caractère

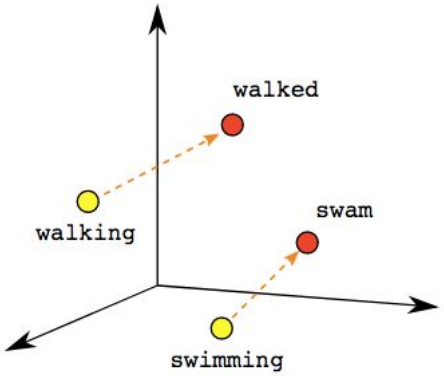
a	0.42	**b**	2.73	**c**	-8.58	**…**
	7.23		7.72		5.69	
	-3.95		-1.60		-1.76	

Ces valeurs sont initialement aléatoires, mais sont “apprises” pendant le processus d'entraînement

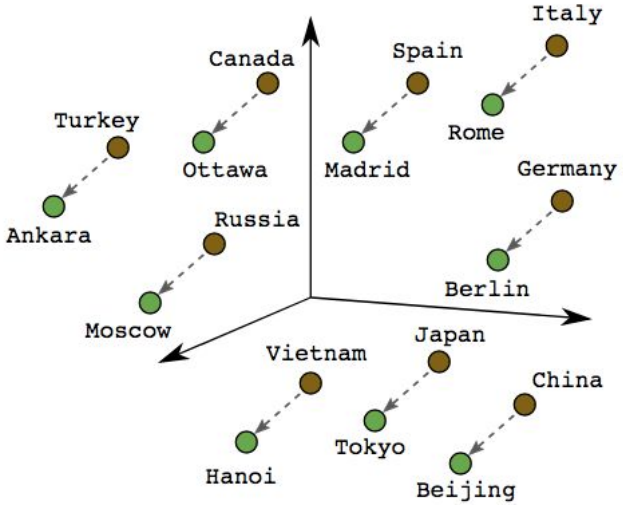
Encodage avec des “embeddings”



Male-Female



Verb Tense



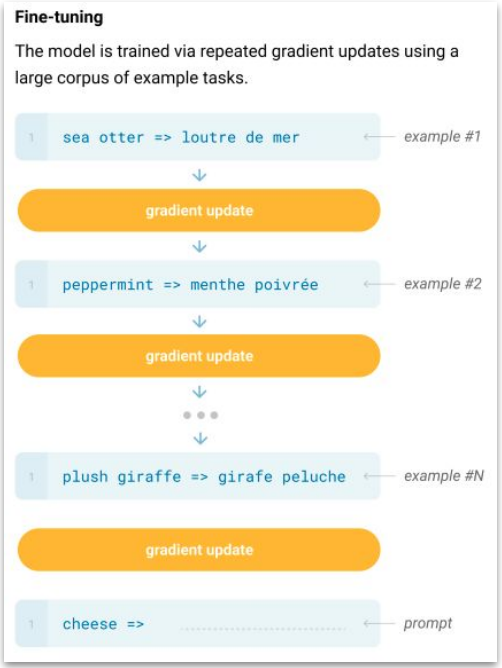
Country-Capital

Traitement du Langage Naturel (NLP)

Comment entraîner un modèle ?

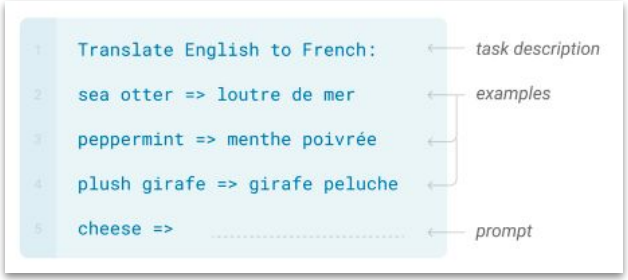
2010-2017: entrainement de zéro
Création d'un nouveau modèle pour chaque tâche

2018: modèles pré-entraînés
Réutilisation de modèles grâce à l'apprentissage par transfert

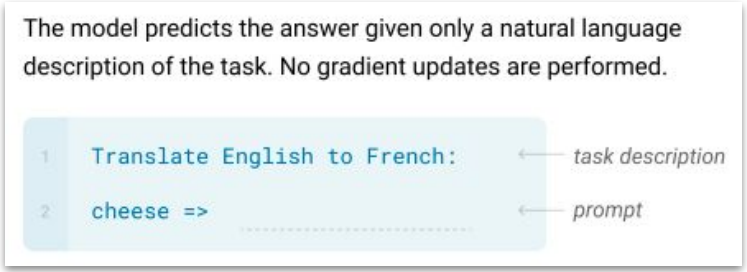


Comment entraîner un modèle ?

2019: apprentissage “few shots”
Création d’un nouveau modèle pour chaque tâche



2020: apprentissage “zero shot”
Réutilisation de modèles grâce à l’apprentissage par transfert

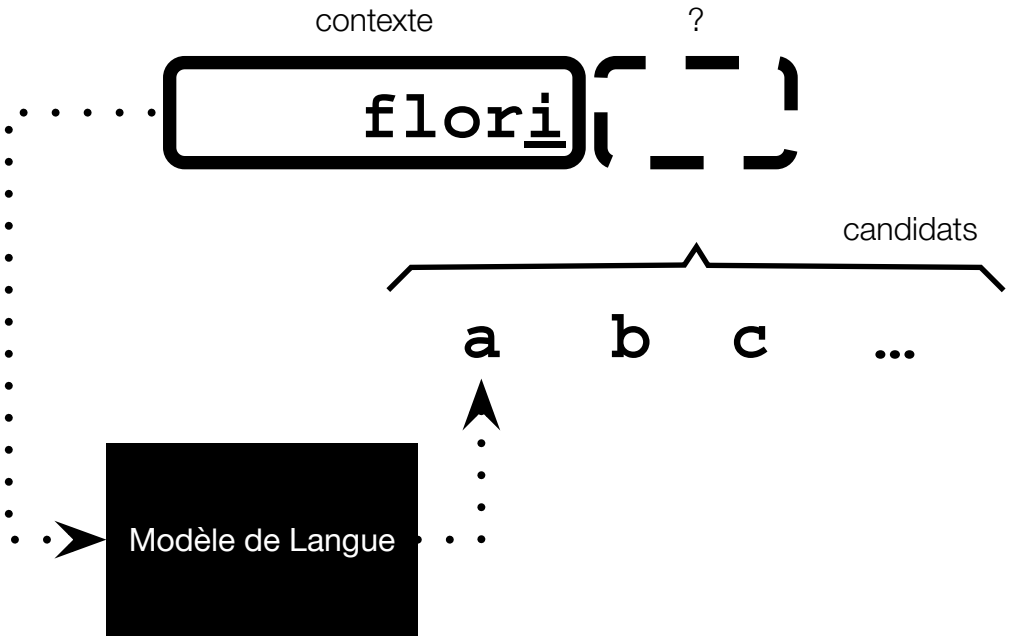


Comment entraîner un modèle ?

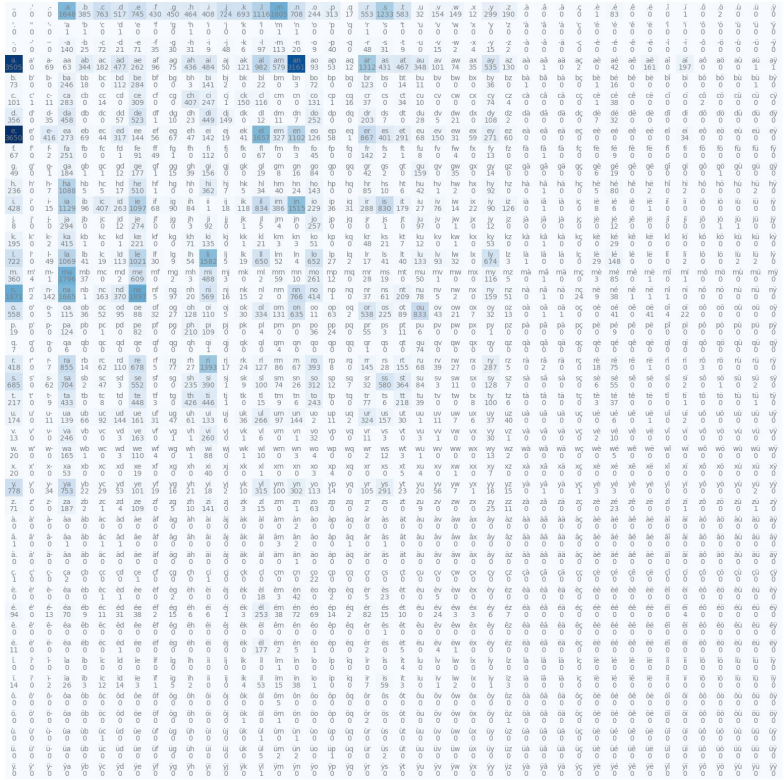
2021: interface conversationnel
Interaction avec le modèle sous forme d'une discussion

2021 : apprentissage "RLHF"
Amélioration du modèle en utilisant du renforcement de l'apprentissage par les retours humains

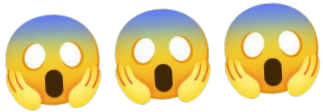
Génération de texte



Contexte plus long



1 caractère : 40 x 40
2 caractères : 1600 x 1600
3 caractères : 64000 x 64000
...
10 caractères : 1.05e16 x 1.05e16



La “malédiction de la dimension”
–Richard Bellman

Solution : réseaux de neurones !

Modèle à 3 couches

"A Neural Probabilistic Language Model", Bengio et al., 2003
https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

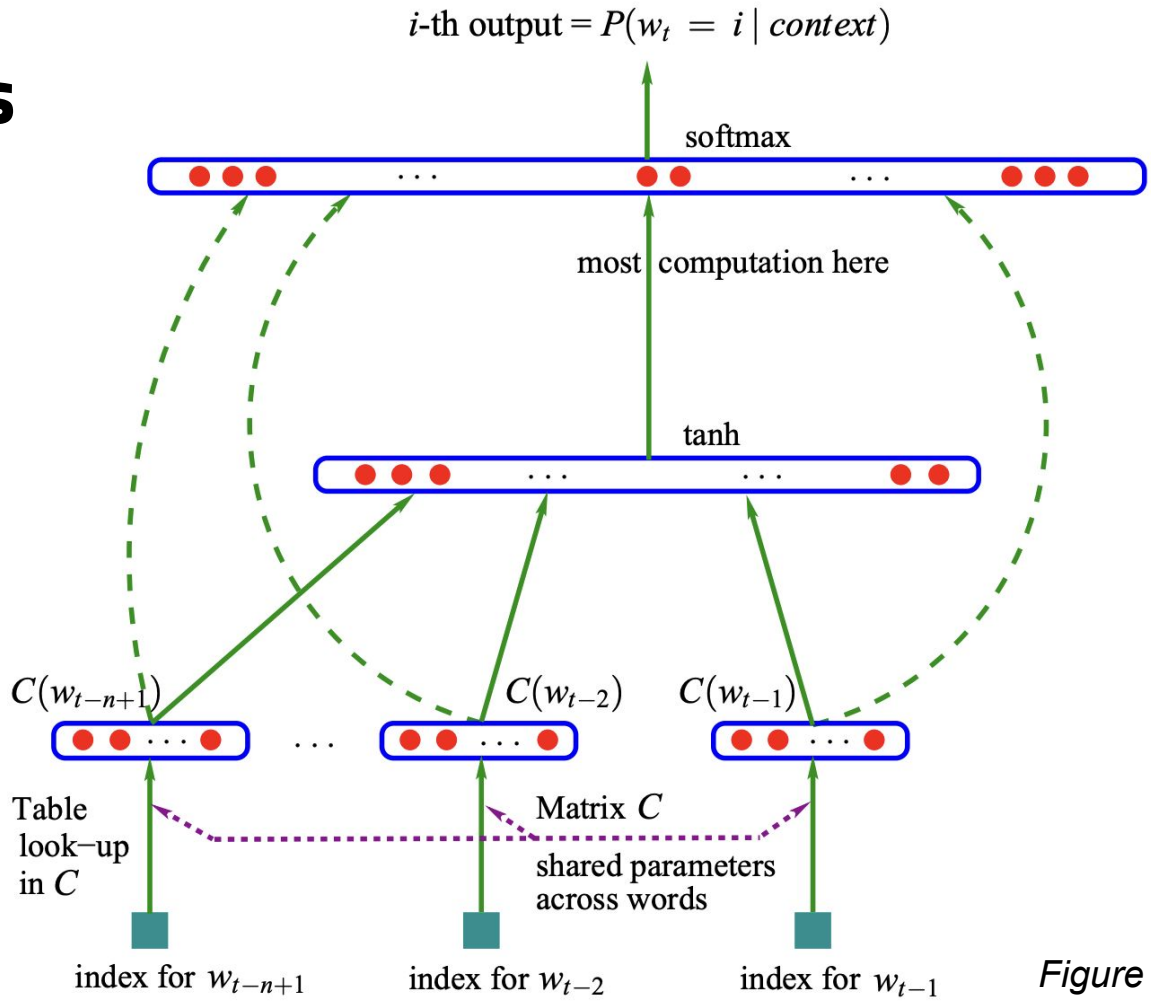


Figure 1

Plus de couches :
Mistral, GPT-4…

```
nb_params = sum(p.nelement() for p in parameters)
print(f"Notre modèle à {nb_params} paramètres")
```

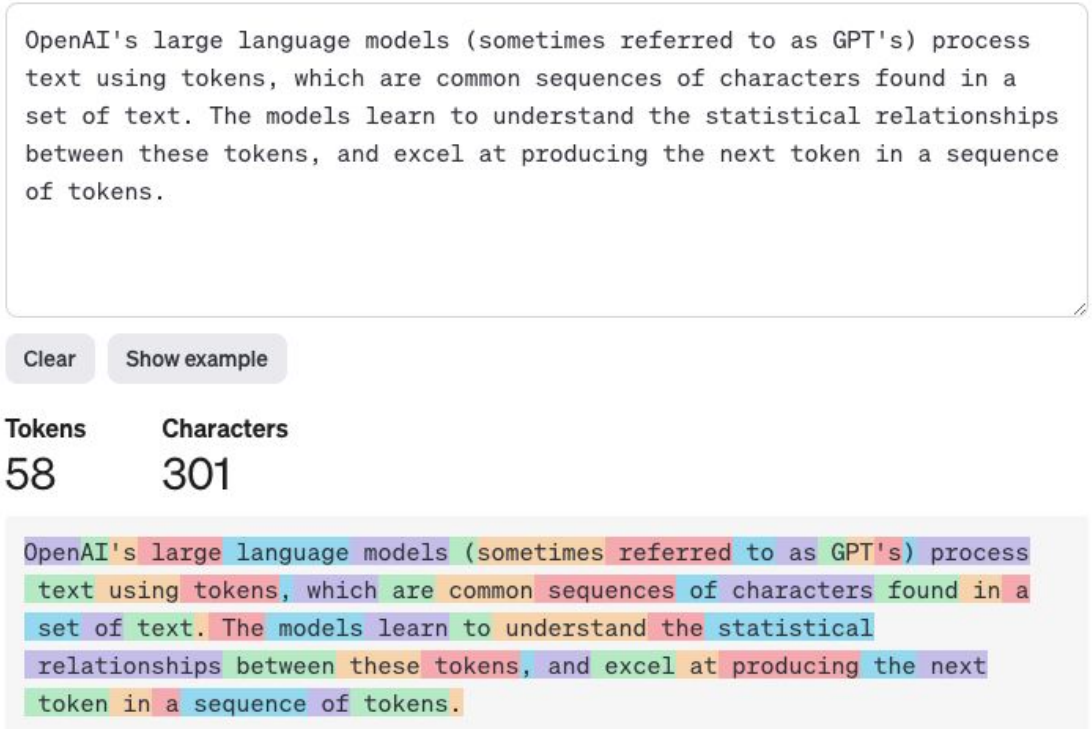
```
Notre modèle à 14007 paramètres
```

LMSYS Chatbot Arena Leaderboard

Rank* (UB)	Model	Arena Elo	95% CI	Votes	Organization	License	Knowledge Cutoff
1	GPT-4o-2024-05-13	1290	+5/-4	22784	OpenAI	Proprietary	2023/10
2	Gemini 1.5 Pro API-0409-Preview	1258	+3/-3	55731	Google	Proprietary	2023/11
2	GPT-4-Turbo-2024-04-09	1257	+3/-3	55236	OpenAI	Proprietary	2023/12
4	GPT-4-1106-preview	1252	+2/-3	77548	OpenAI	Proprietary	2023/4
4	Claude 3 Opus	1249	+3/-2	114090	Anthropic	Proprietary	2023/8
4	GPT-4-0125-preview	1246	+3/-2	70790	OpenAI	Proprietary	2023/12
6	Yi-Large-preview	1241	+4/-4	23822	01 AI	Proprietary	Unknown
8	Bard (Gemini Pro)	1208	+7/-5	11853	Google	Proprietary	Online
8	Llama-3-70b-Instruct	1208	+3/-2	114054	Meta	Llama 3 Community	2023/12
9	Claude 3 Sonnet	1202	+3/-2	90578	Anthropic	Proprietary	2023/8
9	Reka-Core-20240501	1201	+3/-3	34432	Reka AI	Proprietary	Unknown
12	Command R+	1189	+2/-2	57513	Cohere	CC-BY-NC-4.0	2024/3
12	GPT-4-0314	1186	+3/-3	52808	OpenAI	Proprietary	2021/9
12	GLM-4-0116	1184	+6/-6	6990	Zhipu AI	Proprietary	Unknown
12	Qwen-Max-0428	1184	+5/-4	21027	Alibaba	Proprietary	Unknown
14	Claude 3 Haiku	1178	+3/-2	80807	Anthropic	Proprietary	2023/8
17	Qwen1.5-110B-Chat	1164	+4/-4	17836	Alibaba	Qianwen LICENSE	2024/4
17	GPT-4-0613	1161	+3/-3	74440	OpenAI	Proprietary	2021/9

https://chat.lmsys.org/?leaderboard

Tokens



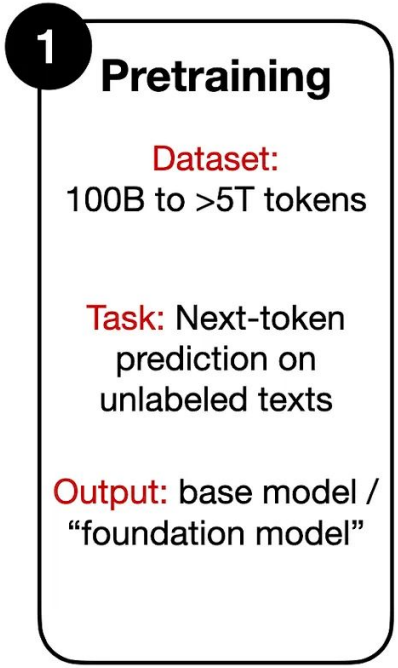
https://platform.openai.com/tokenizer

Mise en pratique

GPT-2

https://colab.research.google.com/drive/1X-TiqsPTNHj7IiNX5DDdUgdLcESOmG7X

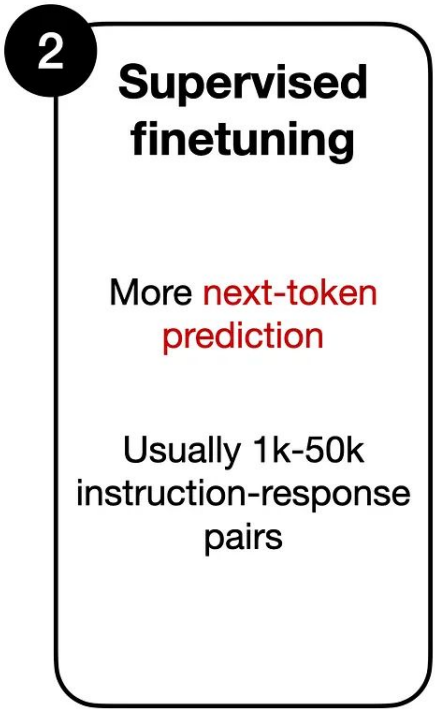
Entrainement en 3 phases



Project Gutenberg (PG) is a volunteer effort to digitize and archive cultural works, as well as to "encourage the creation and distribution of eBooks." It was founded in 1971 by American writer Michael S. Hart and is the oldest digital **library**. Most of the items in its collection are the full texts of books or individual stories in the public domain. All files can be accessed for free under an open format layout, available on almost any computer. As of 3 October 2015, Project Gutenberg had reached 50,000 items in its collection of free eBooks.

https://magazine.sebastianraschka.com/p/llm-training-rlhf-and-its-alternatives

Entrainement en 3 phases




```
{
    "instruction": "Write a limerick about a
                    pelican.”,
    "input": "",
    "output": "There once was a pelican so fine,
              \nHis beak was as colorful as
              sunshine,\nHe would fish all day,\nIn
              a very unique way,\nThis pelican was
              truly divine!\n\n\n"
},
```

```
{
    "instruction": "Identify the odd one out from
                    the group.",
    "input": "Carrot, Apple, Banana, Grape",
    "output": "Carrot\n\n"
},
```

https://magazine.sebastianraschka.com/p/llm-training-rlhf-and-its-alternatives

Mise en pratique

Mistral 7b

https://colab.research.google.com/drive/1YWI7shxHQNszoRuBKIBqHfM08wePpfoP

Entrainement en 3 phases

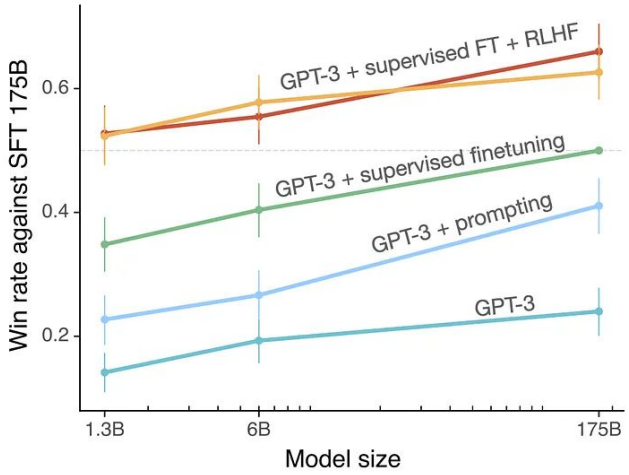
3

Alignment

Align with human preferences

Usually reinforcement learning with human feedback (RLHF)

>50k examples



https://magazine.sebastianraschka.com/p/llm-training-rlhf-and-its-alternatives

GPT-4

Les possibilités de GPT-4

- Résumer
- Traduire
- Expliquer des concepts, enseigner un sujet
- Créer une présentation, un blog post, un tweet
- Critiquer votre document
- Générer des idées
- Expliquer, Générer et déboguer du code
- Coacher
- Classifier
- Convertir non-structuré vers structuré
- Planifier, décider

Des limitations qui disparaissent

- Maintenir la cohérence sur de longs textes (fenêtre de contexte)
- Se souvenir des faits réels / hallucinations
- Tenir compte de l'actualité, accéder au web
- Accéder à vos documents
- Se souvenir des autres conversations
- Utiliser des outils logiciels
- Exécuter du code
- Avoir une personnalité
- Etre formé sur nos documents
- Appeler des APIs externes

Des limitations qui disparaissent

- Maintenir la cohérence sur de longs textes (fenêtre de contexte)
- Se souvenir des faits réels / hallucinations
- ~~Tenir compte de l'actualité, accéder au web~~
- ~~Accéder à vos documents~~
- ~~Se souvenir des autres conversations~~
- Utiliser des outils logiciels
- ~~Exécuter du code~~
- ~~Avoir une personnalité~~
- ~~Etre formé sur nos documents~~
- Appeler des APIs externes

Mise en pratique

GPT-4

https://colab.research.google.com/drive/1VIwC_iOyHtMuUuuErEk1LUnP1ZS89_m7

Projets

Projets

- Présentation finale
  - En personne à l'IFAGE, avec staff ImpactIA
  - Le 25 juin, 18h30 à 21h30
  - Présentation (10min) + questions (5min)
- Le projet
  - Rendu soit le 25 juin, soit jusqu'au 25 juillet
  - 20 à 40h de travail
- Seulement ~~4~~ 5 projets dans #projets 😅
  - Définitions des derniers projets aujourd'hui
  - Travail sur les projets

Démarche

- Commencer par une recherche des outils disponibles
  - On connaît rarement les bons outils dès le départ
  - Être “méfiant” envers les outils et publications
  - Etablir une “baseline” dès que possible
- Commencer simple !
  - Faire le travail à la main, contrôler entrées et sorties
  - GPT-4 est souvent un bon point de départ (même pour les images)
  - Commencer par des outils simple (Keras) avant d’écrire du code PyTorch

Démarche

- Ne rien oublier en route !
  - Garder des notes propres
  - Faire les suivi de ses expériences dans WandB
  - Sauvegarde et backup du code