Demystifying Artificial Intelligence Sorcery

(Part 3: Deep Learning)[a]

Abdelbacet Mhamdi
abdelbacet.mhamdi@bizerte.r-iset.tn

Dr.-Ing. in Electrical Engineering
Senior Lecturer at ISET Bizerte

[a]Available @ https://github.com/a-mhamdi/jlai/

Disclaimer

This document features some materials gathered from multiple online sources.

Please note no copyright infringement is intended, and I do not own nor claim to own any of the original materials. They are used for educational purposes only.

I have included links solely as a convenience to the reader. Some links within these slides may lead to other websites, including those operated and maintained by third parties. The presence of such a link does not imply a responsibility for the linked site or an endorsement of the linked site, its operator, or its contents.

Roadmap

1. An overview
2. Convolutional Neural Network
3. Generative Adversarial Network
4. Variational Autoencoder
5. Natural Language Processing
6. Transfer Learning
7. Reinforcement Learning
8. Quizzes

An overview

CNNs *(Convolutional Neural Networks)* are used for image classification and other computer vision tasks because they are able to automatically learn features from raw data. This is useful for tasks where manual feature engineering is difficult or impractical.

GANs *(Generative Adversarial Networks)* are used for tasks such as image generation and data augmentation because they are able to generate new data samples that are similar to a given dataset.

VAEs *(Variational Autoencoders)* are used for tasks such as image generation and anomaly detection because they are able to learn a compact representation of a dataset and generate new samples from this representation.

NLP *(Natural Language Processing)* is important for tasks such as language translation, text classification, and language generation because it allows computers to process and understand human language.

Convolutional Neural Network

CNN

Motivating Factors

- A **Convolutional Neural Network (CNN)** is a type of neural network that is particularly well-suited for image classification and object recognition tasks. It is designed to process data with a grid-like topology, such as an image,.
- **CNNs** are composed of several types of layers, including convolutional layers, pooling layers, and fully connected layers.

❶ The convolutional layers apply filters to the input data, which are used to detect patterns and features in the data.

❷ The pooling layers reduce the spatial dimensions of the data, which helps to reduce the complexity of the model and make it more robust to small translations of the input data.

❸ The fully connected layers combine the features learned by the convolutional and pooling layers to make a prediction.

CNN

Applications

Image classification **CNNs** can be used to classify images into different categories, such as identifying objects in an image or labeling the scene depicted in an image.

Object detection **CNNs** can be used to detect objects in images or videos and to localize them by drawing bounding boxes around them.

Image segmentation **CNNs** can be used to segment images into different regions, such as separating the foreground from the background or labeling different objects in an image.

Medical image analysis **CNNs** have been used to analyze medical images, such as CT scans and X-rays, for tasks such as tumor detection and segmentation.

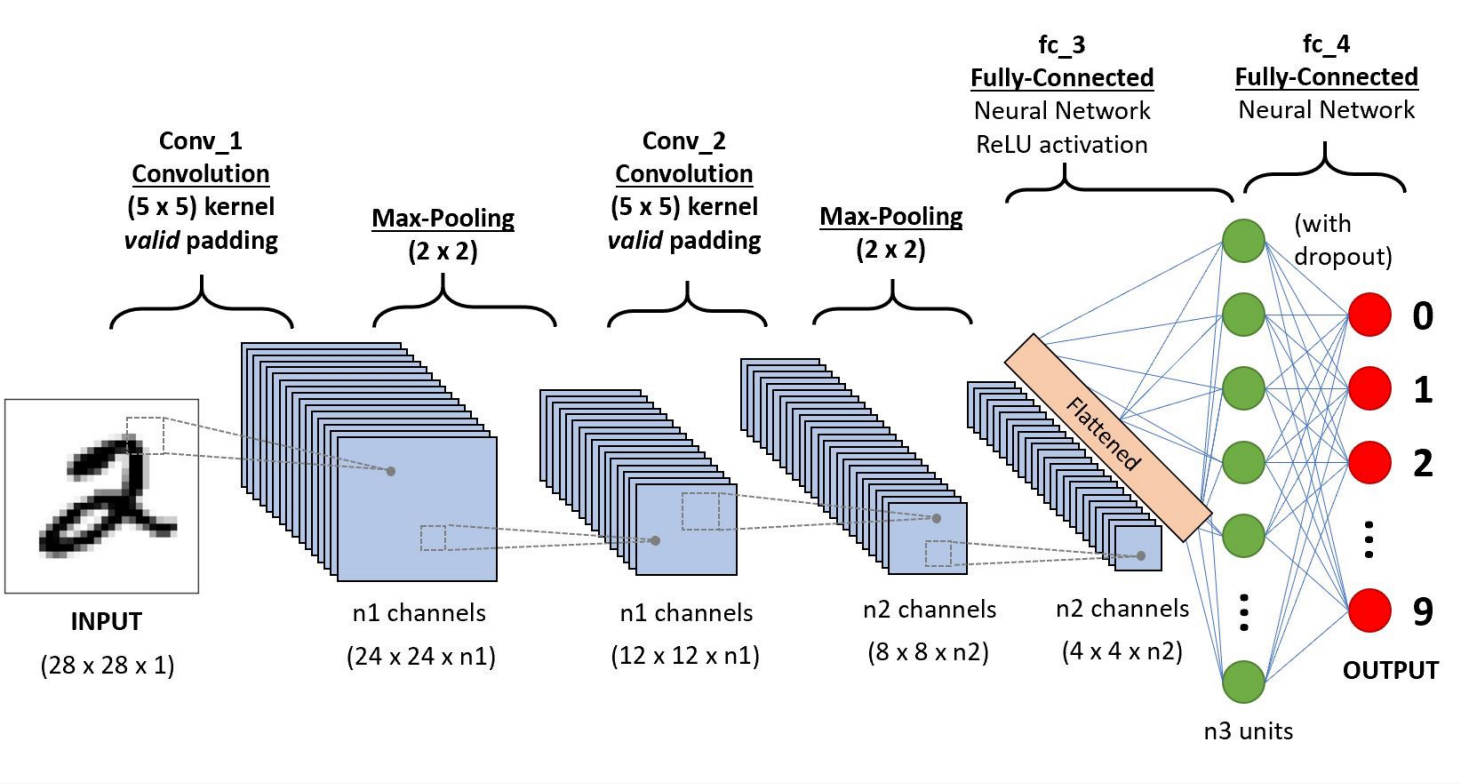
Self-driving cars **CNNs** have been used to process camera images in self-driving cars to detect pedestrians, other vehicles, and traffic signals.

Robotics **CNNs** have been used to process images from robots to enable them to navigate their environment and perform tasks such as grasping objects.

Natural language processing **CNNs** have been used to process text data for tasks such as sentiment analysis and language translation.

CNN

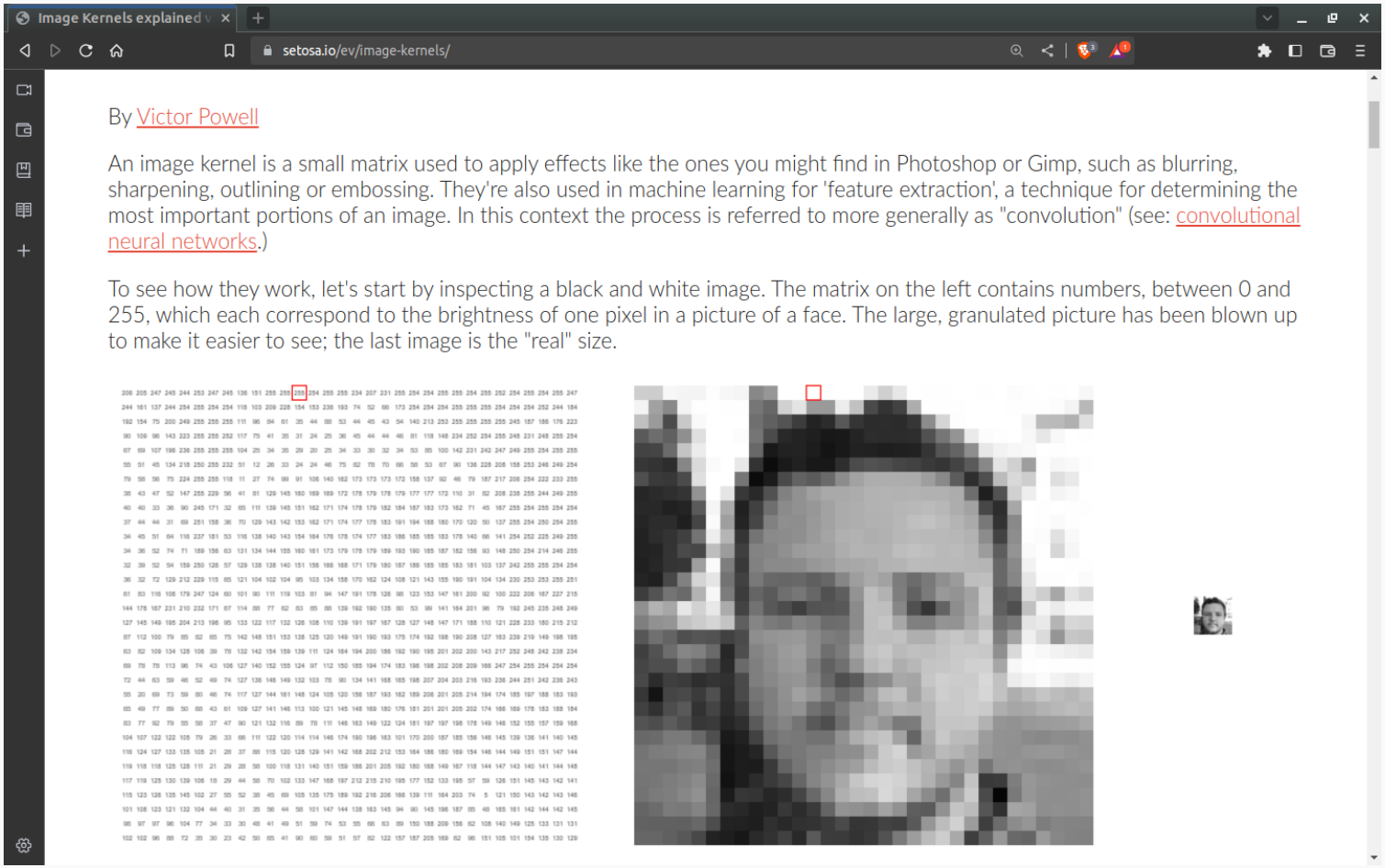
Architecture



▶ Source

CNN

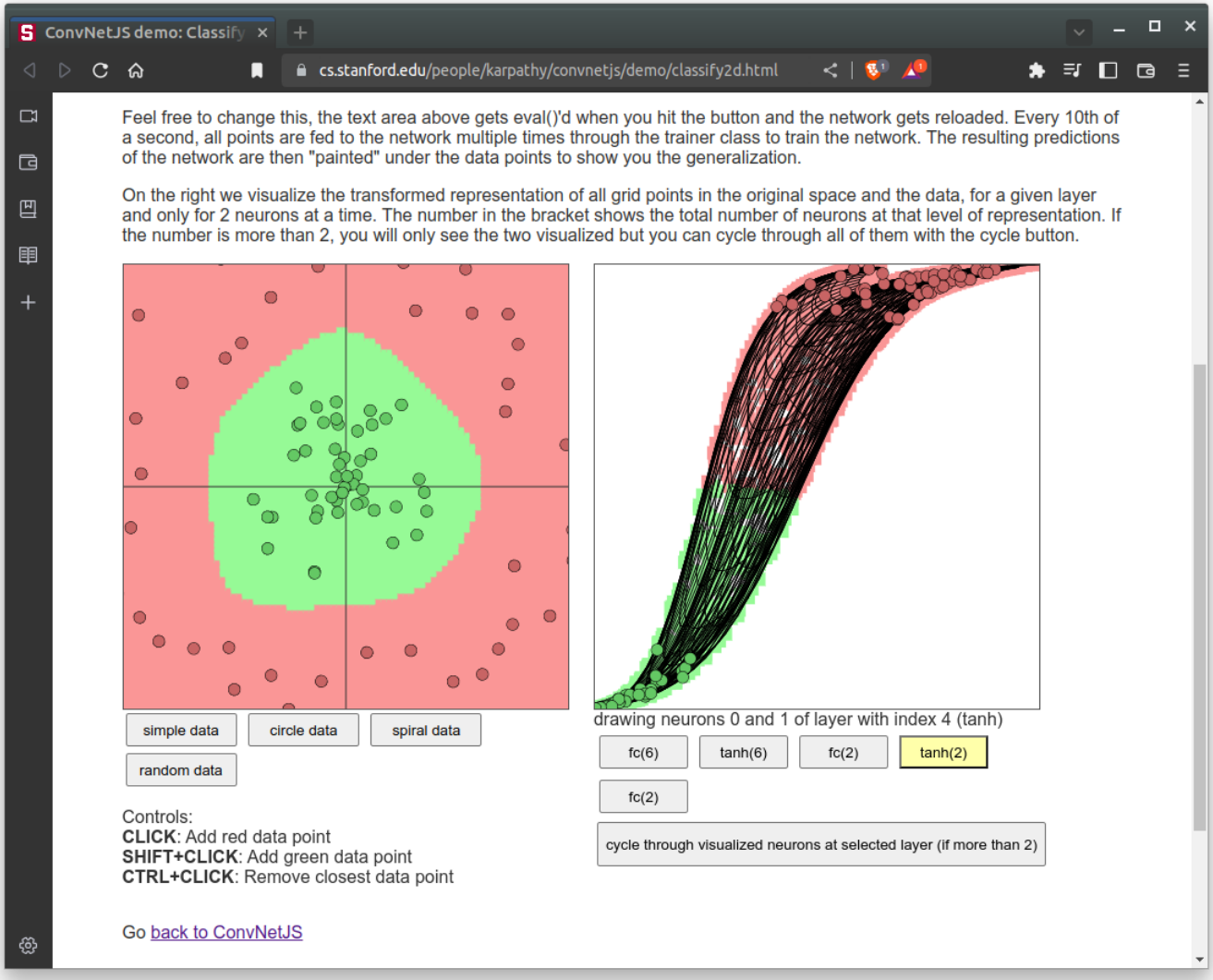
Image Kernels



https://setosa.io/ev/image-kernels/

CNN

ConvnetJS Demo



https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

CNN

Code Snippet

Code is available at https://github.com/a-mhamdi/jlai/
⟶ *Codes* ⟶ *Julia* ⟶ *Part-3* ⟶ *cnn.jl*

Generative Adversarial Network

GAN

An Overview

- A **Generative Adversarial Network (GAN)** is a type of deep learning model designed to generate new, synthetic samples of data. It consists of two networks: a generator network and a discriminator network. The generator network generates synthetic samples, while the discriminator network tries to distinguish between the synthetic samples and real samples of data.
- During training, the generator and discriminator networks are trained concurrently, with the generator trying to generate synthetic samples that are indistinguishable from real samples, and the discriminator trying to correctly classify the samples as either real or synthetic. The generator is trained to improve its synthetic samples based on the feedback from the discriminator, and the discriminator is trained to become more sensitive to synthetic samples.
- The goal of a **GAN** is to learn a generative model that can produce synthetic samples that are similar to the training data. **GANs** have been used for a variety of tasks, including image generation, audio synthesis, and natural language processing.

GAN

Applications

Image generation **GANs** can be used to generate realistic images of objects, people, and scenes.

Image style transfer **GANs** can be used to transfer the style of one image to another image, while preserving the content of the original image.

Text-to-image synthesis **GANs** can be used to generate images from textual descriptions, such as generating images of objects based on their names.

Video prediction **GANs** can be used to predict future frames in a video, given the past frames.

Text-to-speech synthesis **GANs** can be used to generate speech audio from text.

Super-resolution **GANs** can be used to enhance the resolution of images or videos.

Data augmentation **GANs** can be used to generate additional training data for other machine learning models.

Medical image analysis **GANs** have been used to generate synthetic medical images for tasks such as segmentation and classification.

Domain adaptation **GANs** can be used to translate images from one domain *(e.g., synthetic images)* to another domain *(e.g., real images)* to improve the performance of machine learning models.

GAN

Code Snippet

Code is available at https://github.com/a-mhamdi/jlai/
⟶ *Codes* ⟶ *Julia* ⟶ *Part-3* ⟶ *gan.jl*

Variational Autoencoder

VAE

Motivating Factors

- A **Variational Autoencoder (VAE)** is a type of deep learning model that is used to learn latent representations of data. It is a generative model, which means that it can generate new samples of data that are similar to the training data.
- **VAEs** are trained to encode the data into a low-dimensional latent space and then decode the latent representation back into the original data space. During training, the **VAE** learns to reconstruct the input data, while also trying to enforce a constraint on the latent space that encourages it to represent the data in a meaningful way.
- The constraint that is used in a **VAE** is called the variational lower bound. This lower bound is maximized during training, which encourages the latent space to be structured in a way that is useful for generating samples that are similar to the training data.

VAE

Applications

Generative modeling **VAEs** can be used to generate new samples of data that are similar to the training data. This can be useful for tasks such as image generation, audio synthesis, and natural language generation.

Anomaly detection **VAEs** can be used to identify anomalies in data by reconstructing the input data and measuring the reconstruction error. Data points that are poorly reconstructed are likely to be anomalous.

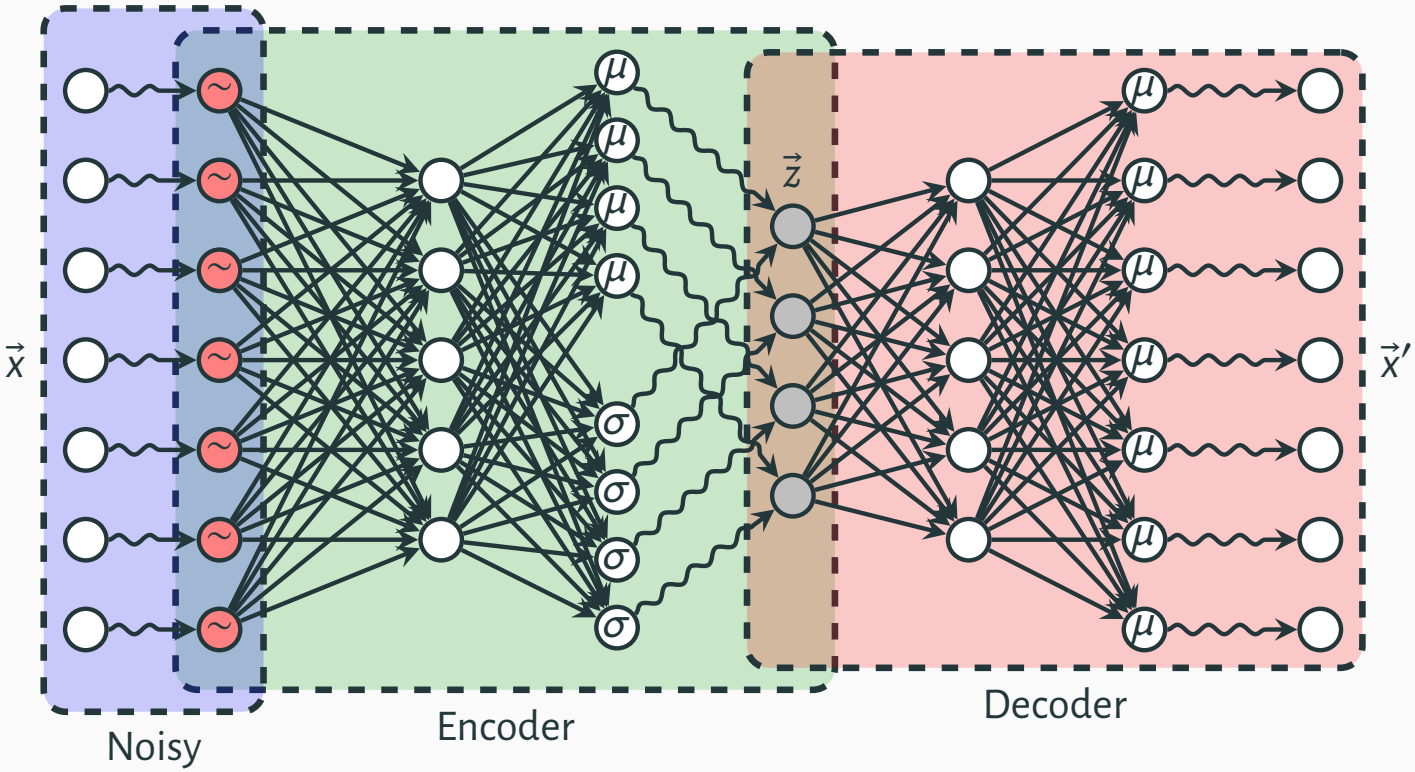
Data compression **VAEs** can be used to compress data by encoding it into a lower-dimensional latent space and then reconstructing it. The encoder can be used as a compression function, and the decoder can be used as a decompression function.

Representation learning **VAEs** can be used to learn meaningful latent representations of data, which can be useful for tasks such as clustering and classification.

Semi-supervised learning **VAEs** can be used in a semi-supervised setting, where only a small portion of the data is labeled. The **VAE** can use the labeled data to learn a meaningful latent representation of the data, and then use this representation to make predictions on the unlabeled data.

VAE

Architecture



VAE

Code Snippet

Code is available at https://github.com/a-mhamdi/jlai/
⟶ *Codes* ⟶ *Julia* ⟶ *Part-3* ⟶ *vae.jl*

Natural Language Processing

NLP

Purpose of NLP

- **Natural Language Processing (NLP)** is a field of artificial intelligence and computer science that focuses on the interaction between computers and humans using natural language.
- **NLP** involves the development of algorithms and models that can understand, interpret, and generate human language.
- **NLP** is used in a wide range of applications, including machine translation, question answering, text summarization, text classification, and sentiment analysis.

NLP

Applications

Part-of-speech tagging Identifying the parts of speech *(e.g., noun, verb, adjective)* in a sentence

Named entity recognition Identifying and labeling named entities *(e.g., people, organizations, locations)* in a text

Sentiment analysis Determining the sentiment *(e.g., positive, neutral, negative)* of a piece of text

Machine translation Translating text from one language to another

Text summarization Generating a concise summary of a longer piece of text

NLP

General Process in Julia

1. Preprocess the text data by lowercasing, removing punctuation, and splitting the text into individual tokens *(e.g., words or subwords)*.
2. Build a vocabulary of the most common tokens in the text data.
3. Encode the text data as a sequence of integers using the vocabulary.
4. Pad the encoded sequences to the same length to make them suitable for input to a model.
5. Define the **NLP** model using a library such as `Flux.jl` or `Knet.jl`.
6. Train the model using gradient descent and a suitable loss function.
7. Use the trained model to make predictions on new data.

NLP

Code Snippet

Code is available at https://github.com/a-mhamdi/jlai/
⟶ *Codes* ⟶ *Julia* ⟶ *Part-3* ⟶ *nlp.jl*

Transfer Learning

Transfer Learning

Driving Forces

- **Transfer Learning** is a machine learning technique in which a model that has been trained on one task is re-purposed on a second related task. **Transfer Learning** can be used to improve the performance of the second task by leveraging the knowledge learned from the first task.
- One common use of **Transfer Learning** is to fine-tune a pre-trained model on a new dataset. For example, a pre-trained image classification model that has been trained on a large dataset such as `ImageNet` can be fine-tuned on a smaller dataset of a different but related task, such as detecting objects in medical images. Fine-tuning the pre-trained model on the new dataset can lead to improved performance compared to training a model from scratch on the smaller dataset.
- **Transfer Learning** is useful because it allows a machine learning model to learn from a large amount of data, even if the data is not directly related to the task at hand. It can also be used to speed up the training process, since the model does not need to be trained from scratch.

Transfer Learning

Applications

Image classification **Transfer Learning** has been used to fine-tune pre-trained image classification models on new datasets, such as identifying plant species from images of leaves or detecting objects in medical images.

Natural language processing **Transfer Learning** has been used to fine-tune pre-trained language models on new tasks, such as sentiment analysis or text classification.

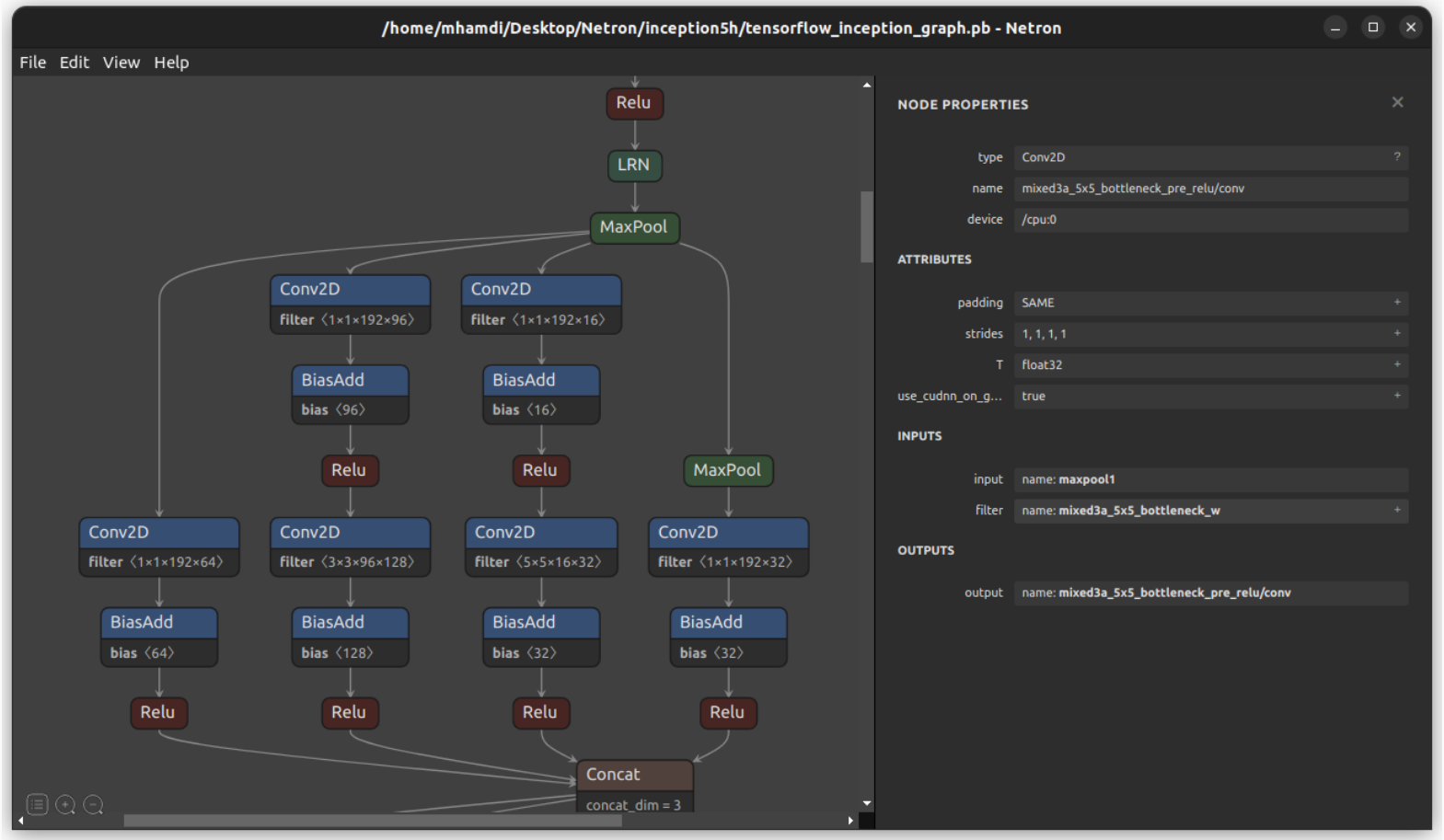
Speech recognition **Transfer Learning** has been used to fine-tune pre-trained speech recognition models on new languages or accents.

Computer vision **Transfer Learning** has been used to fine-tune pre-trained models for tasks such as object detection and image segmentation.

Robotics **Transfer Learning** has been used to fine-tune pre-trained models for tasks such as grasping objects or navigating a new environment.

Transfer Learning

Netron



https://github.com/lutzroeder/netron

Transfer Learning

General Process in Julia

1. Load the pre-trained model *(e.g., a convolutional neural network trained on* `ImageNet`*)*.
2. Replace the final layer (or layers) of the pre-trained model with a new, untrained layer (or layers) that is suitable for your target task.
3. Freeze the weights of the pre-trained layers to prevent them from being updated during training.
4. Load your dataset and split it into training and validation sets.
5. Use the training set to fine-tune the weights of the new layer (or layers) using gradient descent and a suitable loss function.
6. Monitor the performance of the model on the validation set and adjust the hyperparameters *(e.g., learning rate)* as needed.
7. When you're satisfied with the performance of the model on the validation set, you can use it to make predictions on the test set or on new data.

Transfer Learning

Code Snippet

Code is available at https://github.com/a-mhamdi/jlai/
⟶ *Codes* ⟶ *Julia* ⟶ *Part-3* ⟶ *transfer-learning.jl*

Reinforcement Learning

Reinforcement Learning

Synopsis

- **Reinforcement Learning** is a type of machine learning in which an agent learns to interact with its environment in order to maximize a reward. It involves learning to map situations (called states) to actions that will maximize a reward. The agent receives feedback in the form of rewards and penalties for its actions, which it uses to adjust its behavior accordingly.
- In **reinforcement Learning**, the goal is to learn a policy that maximizes the cumulative reward over time. The agent learns this policy through trial and error, by exploring different actions in different states and receiving feedback in the form of rewards or penalties.
- **Reinforcement Learning** is used in a variety of applications, including control systems, game playing, and natural language processing. It has been successful in a number of tasks, including teaching a computer to play chess and Go at a high level.

Reinforcement Learning

Code Snippet

Code is available at https://github.com/a-mhamdi/jlai/
⟶ *Codes* ⟶ *Julia* ⟶ *Part-3* ⟶ *reinforcement-learning.jl*

Quizzes

MCQ (1/1)

1. Your supervisor asks you to create a machine learning system that will help your human resources department classify jobs applicants into well-defined groups. What type of system are you more likely to recommend?
   - × an unsupervised machine learning system that clusters together the best candidates.
   - × you would not recommend a machine learning system for this type of project.
   - × a deep learning artificial neural network that relies on petabytes of employment data.
   - ✓ a supervised machine learning system that classifies applicants into existing groups.
2. Your data science team must build a binary classifier, and the number one criterion is the fastest possible scoring at deployment. It may even be deployed in real time. Which technique will produce a model that will likely be fastest for the deployment team use to new cases?
   - × random forest
   - ✓ logistic regression
   - × KNN
   - × deep neural network
3. The famous data scientist Andrew Ng has been quoted as saying, "Applied machine learning is basically feature engineering." What is feature engineering?
   - × scraping new features from web data
   - ✓ creating new variables by combining and modifying the original variables
   - × designing innovative new user features to add to software
   - × using deep learning to find features in the data

Some Useful Links

1. https://setosa.io/ev/
2. https://karpathy.ai/
3. http://yann.lecun.com/
4. https://www.hackingnote.com/
5. https://machinelearningmastery.com/
6. https://stanford.edu/ shervine/teaching/
7. https://www.ibm.com/downloads/cas/GB8ZMQZ3
8. https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

References

[Alz+21] L. Alzubaidi et al. “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of Big Data* 8.1 (2021). DOI: 10.1186/s40537-021-00444-8.

[Azu21] P. Azunre. *Transfer Learning for Natural Language Processing*. Manning Publications Co. LLC, 2021, p. 272.

[Goo+17] I. Goodfellow et al. *Deep Learning*. MIT Press, 2017.

[HZM16] X. Hao, G. Zhang, and S. Ma. “Deep Learning”. In: *International Journal of Semantic Computing* 10.03 (2016), pp. 417–439. DOI: 10.1142/s1793351x16500045.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[LD19] B. Lauwens and A. B. Downey. *Think Julia : How to Think Like a Computer Scientist. How to Think Like a Computer Scientist*. O’Reilly Media, 2019, p. 298.

[PWP20] D. Phil Winder Ph. *Reinforcement Learning Industrial Applications of Intelligent Agents. Industrial Applications of Intelligent Agents*. O’Reilly Media, Incorporated, 2020.

Further Reading (2/2)

[Sar21] I. H. Sarker. “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions”. In: *SN Computer Science* 2.6 (2021). DOI: 10.1007/s42979-021-00815-1.

[SB] R. S. Sutton and A. G. Barto. *Reinforcement Learning An Introduction. An Introduction*. A Bradford Book, p. 552.

[SC21] F. F. L. da Silva and A. H. R. A. H. R. Costa. *Transfer Learning for Multiagent Reinforcement Learning Systems*. Springer International Publishing AG, 2021.

[Ser21] L. Serrano. *Grokking Machine Learning*. Manning Publications Co. LLC, 2021, p. 498.

[SKP] M. Sewak, M. R. Karim, and P. Pujari. *Practical Convolutional Neural Networks: Implement advanced deep learning models using Python*. Packt Publishing - ebooks Account, p. 218.

[SR] J. Silge and D. Robinson. *Text Mining with R: A Tidy Approach*. O’Reilly Media, p. 194.