AI for the Arts and Humanities (B)

Work Book

# README

The first section of this workbook on **Getting Started** can be skipped if you have already carried out the tasks in the [lab sheet for Week 1](https://moodle.gla.ac.uk/mod/resource/view.php?id=4099452). This section summarises the key new tasks for this semester from the Week 1 lab sheet. If you have done Week 1 tasks, you can move onto **Exploring AI Tools**.

The plan is to incorporate all tasks and any edits of tasks into this one sheet as the course progresses. The work book will be linked to each week but it will be the same workbook being linked but changing to reflect additional content and error corrections!

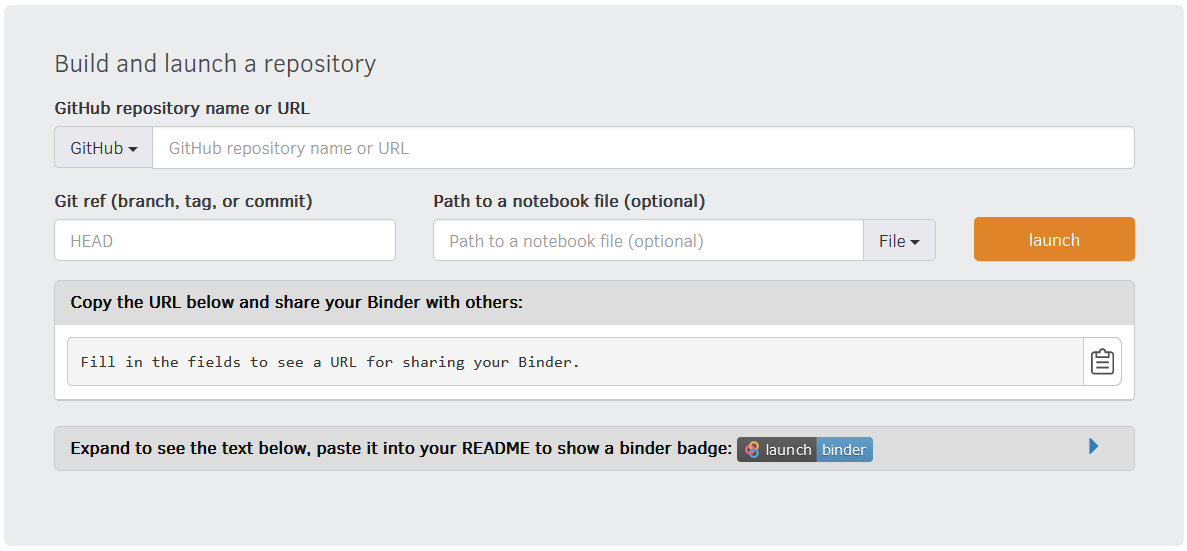
# Getting Started (Week 1) – summary included for completeness. Lab task also included in Week 1.

1. Set up a new GitHub Repository for this semester’s course AI for the Arts and Humanities (B). You learned how to set up a repository in course (A).

If you are not yet familiar with setting up accounts and repositories on GitHub, the following can be useful.

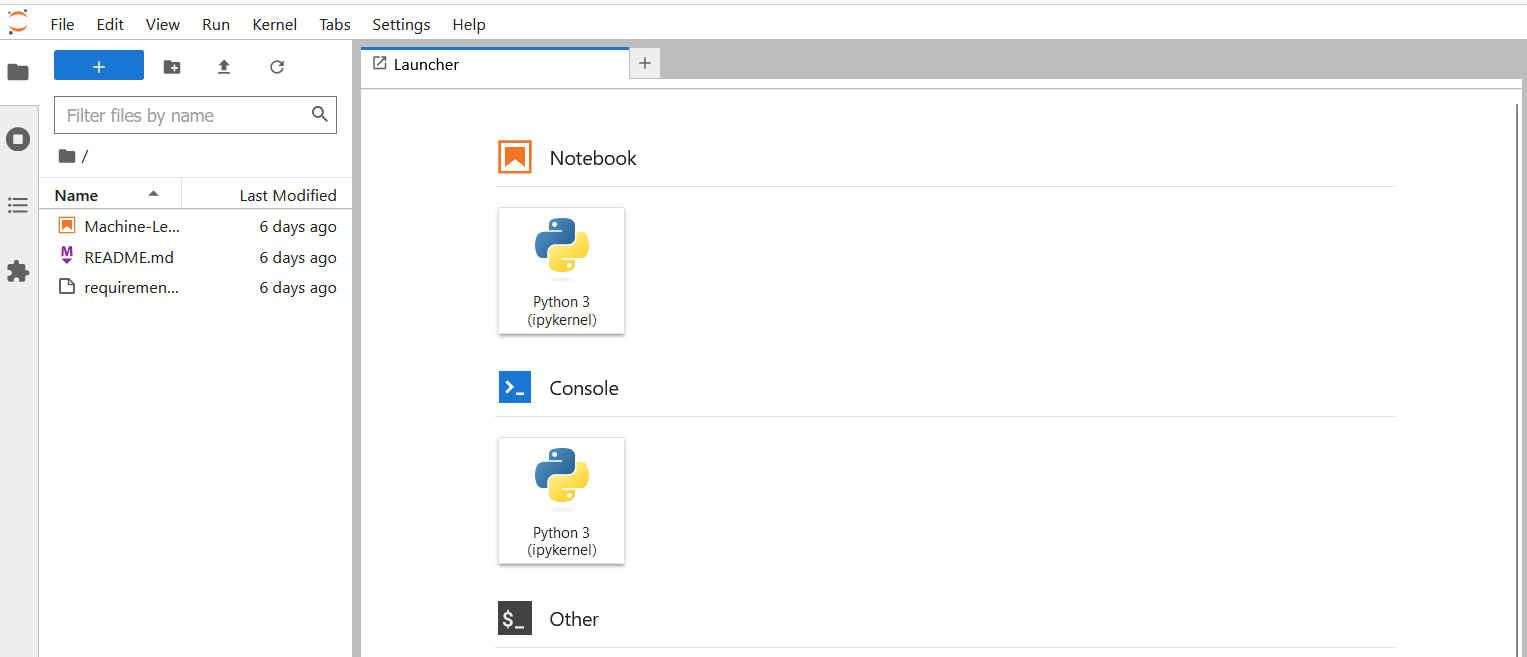
* + 1. **GitHub pages showing you the basics**. Even if you’ve done it in course (A), these pages have more to learn (e.g. managing branches):
       - Creating an account: <https://docs.github.com/en/get-started/quickstart/creating-an-account-on-github>
       - Hello World: <https://docs.github.com/en/get-started/quickstart/hello-world>
         1. How to create a repository
         2. Starting a managing a new branch: think of those time travel movies you’ve watched! This is like creating a new time line.
         3. Making and committing changes – this “pushes” content to the GitHub
         4. Merging changes on different branches – this “pulls” content from other branches into the current branch.
    2. **Talk to your peers who might have become more familiar.** 
       - We all come from different backgrounds in this course – this can be both a challenge and an opportunity!
         1. Get advice
         2. Have your peers check your URL

1. **Launching your GitHub repository as a collection of interactive notebooks.**
2. Create a requirements.txt file for one of your notebooks from last semester.
3. Upload both your notebook and requirements.txt to your new GitHub Repository. Commit your changes on GitHub.
4. Navigate to <https://mybinder.org>
5. Paste in your Repository URL into the top box labelled **GitHub repository name or URL**
6. Type in your branch name in the box labelled **Git ref (branch, tag, or commit)** – if you’ve not created any branches, this would be **main**

****

1. Make a note of your URL to share with peers and tutors in the box labelled **Copy the URL below and share your Binder with others.**

Clickthe button labelled **Launch.** Once it is finished, you will see a window with your notebooks listed in the left hand column and links for creating new notebooks in the main section. This is an instance of **Jupyter Lab**.



1. **Check your notebook:** double click on your notebook name on the lefthand side to start it up.
   1. If your requirements.txt listed all your libraries/packages correctly, then you will be able to run all your code cells without any problem. Try running your cells!
   2. If you get an error then your requirements file did not list all the packages or they were not successfully installed in the Binder Jupyter lab instance.
2. **Editing anything in this Jupyter Lab environment does not change anything on your GitHub repository, so if you make changes and quit, you will lose your changes!**
3. If you want to save your changed notebooks, you need to download and upload to GitHub. It is intentionally design this way for security reasons!
4. [Share your GitHub repository URL and Binder URL](https://moodle.gla.ac.uk/mod/url/view.php?id=4218423) – link on Moodle for Week 1.
5. **(Optional)** We did not do this in course (A), but you can set up Git on your local machine. This allows you to “push” changes to your GitHub from your machine directly (instead of uploading or editing at the website).
6. Instructions here: <https://docs.github.com/en/get-started/quickstart/set-up-git>
7. YouTube Tutorial from netninja: <https://www.youtube.com/watch?v=3RjQznt-8kE&list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR>
8. **(Optional)** In course (A), we used Anaconda Navigator to set up our environment and to install libraries and packages. You can do all of this on Anaconda prompt.
   * **Instructions are included in Week 1 lab of the current course (B)**.
   * **Note:** use pip sparingly for installing your libraries!! **Unlike conda, pip does not collect and resolve all the dependencies to find associated packages and correct versions before installing.**

# Exploring AI Tools (Week 2)

## 1. Generative AI Tools

### Task 1.1: Read this Introduction

Later in the course, you will have an opportunity to attempt more Python coding, for example, to make use of pretrained models. For the exercises in this lab, however, we are going to explore some well known ready-made generative AI tools that you can use without coding.

There are many tools out there, and it will not be possible to explore all of them in this lab. The aim of this lab is to get a flavour for what is available out there. You will need to spend some time exploring the space on your own. Feel free to share the tools you discover with the class on the [AI Q & A Forum](https://moodle.gla.ac.uk/mod/hsuforum/view.php?id=3447898).

Note:

* Some tools, while free to use, require that you create an account. [ChatGPT](https://openai.com/chatgpt) from OpenAI is one of these tools. The updated version [ChatGPT Plus](https://openai.com/blog/chatgpt-plus) is not free! These tools often require you to give away your information and data – not only at the time of registration but during use.
* Some tools also limit your credits (e.g. DALL-E 2, ArtsBreeder etc.) and/or allows only limited free credits to be used within a set time period, This applies to OpenAI tools such as GPT-3 Playground as well.
* Many tools with Google (e.g. [Google Bard](https://bard.google.com)), NVIDIA, and/or IBM are also depend on creating an account, even when it is free.
* You should have some limited access to [Microsoft Copilot](https://copilot.microsoft.com/) using your UofG account. This provides you with some protection of your personal and organisational data. However, always be careful of any limits imposed or conditions of use.
* There are also tools like [Stable Diffusion Online](https://stablediffusionweb.com/), which seem like it is free but also collects your information. You should not only look at the main page for permissions but look out for links that lead to permissions regarding “vendors”.
* Many of these tools are currently attracting much attention as state-of-the-art in generative AI, be cautious on who you decide to share your information with to create an account.

The example described in this lab use tools that can be used off the shelf (e.g. Hugging Chat, limited free credit on ArtBreeder). You task can, however, be carried out with any number of tools and we will leave you to decide whether you want you sign up for tools with OpenAI and/or Google, with the understanding that you do so at your own risk.

* As students of digital media and information studies, you should ask yourself what **risks there are regardless of whether you create an account or not**, as you are giving away data. It is important to sign up and/or use tools with caution.

**Recall that in this course, we are not pushing the boundaries to develop the best AI performance.** Rather we are exploring how these tools augment the human creative process, that is, we are assessing how human-AI collaboration can achieve more than either on their own.

### Task 1.2: Read/Watch the following

* ["AI Should Augment Human Intelligence, Not Replace It"](https://hbr.org/2021/03/ai-should-augment-human-intelligence-not-replace-it) (2021)
* [“Don't fear intelligent machines. Work with them”](https://www.ted.com/talks/garry_kasparov_don_t_fear_intelligent_machines_work_with_them) (2017)
* [“The rise of human-computer cooperation”](https://www.ted.com/talks/shyam_sankar_the_rise_of_human_computer_cooperation) (2012)

Ask yourself the question: **do** **the best AI and best human lead to the best collaboration**? What makes a good collaboration (human or AI)? Share your thoughts with your peer group to get a feel for existing views.

## 2. Working with cultural heritage

For your assessment for this course you will be expected engage with, and engage others with, cultural heritage objects. In this task you will experiment with different ways you might use AI tools to help you.

In the exercise below, we will focus on medieval texts. This is for illustrative purposes only. You can choose another subject if you like. You can focus on something you know or browse the [National Museum of Scotland database](https://www.nms.ac.uk/explore-our-collections/search-our-collections/) for a topic of interest. You could select your subject from [Wikipedia](https://en.wikipedia.org).

### Task 2.1: Review the following example idea

#### Identify your topic and value

The subject of interest for me showcased in the example below is Cyfraith Hywel - the topic of [Zoe Bartliff](https://www.gla.ac.uk/schools/humanities/staff/zoebartliff/)’s PhD thesis. Cyfraith Hywel is a early tradition underpinning the legal system and cultural identity of Wales. It is not widely known despite its importance. Engaging the public with this resource will enhance a rich understanding of Welsh culture.

#### Explore the background of your topic

1. **What is it?**

As a starting point I asked Hugging Chat: **“What is Cyfraith Hywel?”**

**Response:**

Cyfraith Hywel (also known as Welsh law or Law of Howell) refers to the legal system that existed in Wales prior to its conquest by England in the late 13th century. It was named after King Hywel ap Cadell, who ruled over much of Wales during the early 10th century and played a significant role in codifying existing laws and customs.

Cyfraith Hywel consisted primarily of customary laws governing matters such as property rights, inheritance, marriage, theft, homicide, and compensation for injuries. Many of these laws were influenced by earlier Celtic traditions and practices, although Christian principles and Roman law also had an impact. The laws were originally transmitted orally, but later written records emerged in the form of manuscripts containing collections of legal maxims, cases, and commentaries.

After the Norman Conquest of England in 1066, English influence gradually increased in neighboring Wales, leading to the gradual erosion of Cyfraith Hywel. However, elements of the traditional Welsh legal system persisted until the Act of Union in 1536, which formally incorporated Wales into the Kingdom of England. Today, Cyfraith Hywel remains an important part of Welsh cultural heritage and continues to inspire scholarly research and debate.

1. **Fact Checking**

Generative AI is prone to delivering inaccurate information so it is important to check the validity of the AI’s response. Checking validity is for everyone – AI or human – not all published content is accurate or precise. While it is not necessarily a reliable source, as a first port of call, you might look up information in Wikipedia as well as other scholarly sources. Much of the above seems in alignment with the Wikipedia page.

1. **Narrowing down your focus**

In my experience, medieval texts were often distinguished by their illumination. This perception led me to explore **illuminated** manuscripts that might be relevant to Cyfraith Hywel. Three items were identified by Hugging Chat:

* **Llyfr Colan (Book of Colan):** An early 14th-century manuscript containing sections of Cyfraith Hywel, along with religious texts and poetry. Its pages feature elaborate initial letters, decorated borders, and illustrations depicting scenes from Welsh mythology and history.
* **Peniarth MS 28 (Black Book of Chirk):** Another 14th-century manuscript that contains extensive extracts from Cyfraith Hywel, alongside historical chronicles and genealogies. Its pages feature intricate penwork decoration, including stylized animals, foliage, and geometric patterns.
* **Llyfr Coch Hergest (Red Book of Hergest):** A large compendium of Welsh literature and scholarship produced in the late 14th century, which includes a section devoted to Cyfraith Hywel. Like other Welsh legal texts, it features colorful initial letters and decorative flourishes.

1. **Fact Checking**

It was necessary to check whether the manuscripts identified by Hugging Chat really exist and are associated with Cyfraith Hywel. The only item I could verify as actually associated with Cyfraith Hywel is Peniarth MS 28, but even this is not the same as Black Book of Chirk, which is Peniarth 29. This led me abandon Hugging Chat’s to explore the items at this Wikipedia page on [List of Cyfraith Hywel Manuscripts](https://en.wikipedia.org/wiki/List_of_Cyfraith_Hywel_manuscripts) and more specifically this item: <https://en.wikipedia.org/wiki/NLW_MS_20143A>. Exploration of the National Library of Wales page for this item shows that “it includes many interesting illustrations, including a two-headed dragon, a mermaid and all kinds of animals”. The page says this is very unusual for a text of Cyfraith Hywel.

#### Experimenting with AI generated image

I first identified an illustration from the manuscript NLW MS 20143A. Then I used free credits on [Artbreeder Prompter](https://www.artbreeder.com/create/prompter)to generate an image similar to the illustration to create another illustration. I have presented the illustrations below:

A red fish with a pattern on it

Description automatically generated with medium confidenceA red dragon with wings on a brown surface

Description automatically generated

Which do you think was illustrated by AI? I might explore how well this might work with [Arbitrary Style Transfer](https://reiinakano.com/arbitrary-image-stylization-tfjs/) (completely free) instead of ArtBreeder (only three credits allowed per month).

#### Identifying Some Preliminary Objectives

Cyfraith Hywel is a legal system and its text is not represented by one but 80+ manuscripts and derivatives. To engage with this item and engage others with this item, I would like to:

* + - Find a data visualisation method to demonstrate the scope of the 80+ manuscripts.
    - I would like to use generative AI to engage the audience with illustrations found in selected manuscripts by challenging them to scrutinise to distinguish between human vs AI generated content.
    - This will inspire audiences to appreciate Welsh history and question the notion of authenticity of archival material.

### Task 2.2: Now It’s Your Turn!

* Explore Wikipedia to identify your topic. It could be about a historical figure, a famous cultural artifact, or a scientific concept.
* Explore the background of your identified topic. You can do this manually or get the help of AI tools.
* Experiment with Generative AI tools. I used image generation above but yours can be text generation, audio generation or even video generation or beyond.
* Identify some initial objectives/ideas. Generating images and exploring the background of the topic I selected allowed me to understand the scope of relevant areas and possible avenues for engaging the public and for learning and discovery! Do the same with your topic.

### Task 2.3: Explore Other Tools

There are also online tools you can explore.

Tools that usually work with pre-trained models:

* [Arbitrary Style Transfer](https://reiinakano.com/arbitrary-image-stylization-tfjs/) to apply style transfer to an image. This take a content image and style image to produce the content of the content image in the style of the style image.
* [DeepDream](https://huggingface.co/spaces/unnati026/DeepDream) to create a dream-like effect on a given image. These extract features using the layers of a pretrained model to overlay these onto an existing image.
* [Stable Video Diffusion](https://stablediffusionweb.com/Stable-Video-Diffusion) (to create videos from images) and [Meta’s Audiobox](https://audiobox.metademolab.com/maker) to generate synthesised voice and sound effects.
* Search for your own tools on Hugging Face and beyond. share your tools on the Moodle. Some examples:
  + [Image captioning tool](https://huggingface.co/spaces/nielsr/comparing-captioning-models) to create textual descriptions of the image(s). An alternative [Stable Diffusion 2 implementation](https://huggingface.co/spaces/stabilityai/stable-diffusion) to generate more images.
  + The [Riffusion real-time image-to-music generation](https://huggingface.co/spaces/ybelkada/image-to-music) to create music for your images.
  + You can also use textual prompts to generate music with [Riffusion real-time music generation](https://huggingface.co/spaces/fffiloni/spectrogram-to-music).

## 3.Reflection

### Task 3.1 Assessing AI performance and human intervention

What is your assessment of the quality of AI output in terms of reliability? For example, do you agree with the captions provided for the images? Do different algorithms perform differently? Likewise, what advantages and disadvantages does ArtBreeder have in comprison to tools like Stable Diffusion?

Most importantly, how does human intentionality, agency, and labour, for example, as curator and director, allow enhancement of the creative process? Go back to the lecture in Week one to reflect on the **Lovelace-Boden-Ridler Test**. What change would be required in the process to pass the test?

These tools can be applied multiple times in a sequence to apply the transformations again on the output of the previous step - why would you do that and what could that reveal?

### Task 3.2 New perspectives and challenging status quo

In addition to the questions in the previous section,

* How can the collaboration between AI and human be designed to question the way your chosen Wikipedia article is presented?
* How can AI and humans collaborate to introduce a new perspective that did not emerge from the article - e.g. (as a suggestion) about underrepresented groups in historical narrative?

### Task 3.3 Examples from AI Artists

Explore how Marion Carre in her talk questions the notion of **Trust** in the archive. Watch the video ["New Forms of Togetherness"](https://www.youtube.com/watch?v=XVRDRyqSuL8) to see how Marion questions the concept of "the real" and "the fake" in the archives. Marion will be joining us in Week 8 to talk more about her work.

Explore [Alex Reben's work](https://www.youtube.com/watch?v=ayZ0n2vEuPI), in particular, the variety of ways content generation of many types can be combined to produce creative work. The presentation illustrates the importance of humans as curator, archivist, and project manager in the creative process.

Discover how Nigerian AI artist Malik Afegbua is making a difference by ["reimagining a stylish old age"](https://www.the-star.co.ke/news/africa/2023-01-15-the-nigerian-ai-artist-reimagining-a-stylish-old-age/). Likewise, think about how you can influence the perceptions of society through AI and creativity.

Reflect on AI and creativity not just as a form of AI art but as a force for good by raising new perspectives and inviting others to question how we represent people, culture and scientific concepts. How would you incorporate that into your projects?

## 4. Python Homework

Next week, we will get stuck back into a little bit of Python code to understand some of the things going on under the hood.

You might benefit from revising some of the programming concepts we learned and augment your knowledge on new concepts (e.g. how to define a function and what it is), go through the Kaggle tutorials ["Intro to Programming"](https://www.kaggle.com/learn/intro-to-programming) and “[Python](https://www.kaggle.com/learn/python)”.

Because we did little bits and pieces in course (A), you will encounter familiar concepts (e.g. variables, lists, data types), but the Kaggle course will consolidate and extend your knowledge.

**(Optional)** If you want to extend your knowledge even further, you can attempt the tutorial at [learnpython.org](https://learnpython.org/). However, **note that learnpython.org makes a few basic assumptions about your experience with programming** so it is best to start with Kaggle, unless you have programmed before!

# Exploring Generative Models – Part I (Week 3 - 4)

In this section we will explore some generative models in AI. You’ve already explored some last semester in your assessment work. This semester we will look at a few more. We will start with models that use pre-trained models as a leverage: Deep Dream and Neural Style Transfer.

Later in the course we will selectively explore four of the foundational models of generative AI: variational autoencoders, transformers, stable diffusion and generative adversarial networks. Most state-of-the-art approaches currently used in applications are a variation and/or combination of these models. The terminology can be inscrutable at the beginning – but don’t worry! You will start to get the idea as the course progresses.

Before we begin, let’s review some Python to help you build on the skills from last semester and enhance your code reading ability.

## 1. Python Revisited

Last semester, you were asked to follow along some Python code (almost blindly!).

* The aim was to get you used to some standard concepts in programming, such as importing libraries and packages, assigning values to variables, and manipulating those variables.
* We also encountered a variety of data structures (for example, lists, dictionaries and arrays), and took a peak at some pre-trained models (such as, the convolutional neural network model MNIST for recognising hand-written digits, and Word2Vec word embedding models).
* We further looked at a number of machine learning libraries (e.g. Scikit-Learn a.k.a sklearn) to understand the fundamental stages of machine learning. For example,
  1. Loading data
  2. Splitting the data into training and testing data
  3. Choosing and setting up the model
  4. Fitting the model to the training data

To get you back started on Python and to help you consolidate some programming concepts and extend on them, at the end of the last lab, we suggested you:

### Task 1.1: Go through the [Kaggle’s Intro to Programming](https://www.kaggle.com/learn/intro-to-programming)

If you did not do this, we strongly recommend you start with this course before you do the other exercises in this lab. It will help you understand the technical discussions to follow. First focus on the first three sections ([Arithmetic and Variables](https://www.kaggle.com/code/alexisbcook/arithmetic-and-variables); [Functions](https://www.kaggle.com/code/alexisbcook/functions); [Data Types](https://www.kaggle.com/code/alexisbcook/data-types)) of the Kaggle tutorial.

**You do not need to go through the whole tutorial in one sitting!**

### Task 1.2 Quick Quiz

1. According to the Kaggle tutorial section on [*Data Types*](https://www.kaggle.com/code/alexisbcook/data-types), what types are the values assigned to variables in the image below? Create a code cell in your notebook to check by using the *type* command and *print* command. And add a comment in the cell to give the answers.

A screenshot of a computer

Description automatically generated

1. In course (A) (Weeks 1-3), we saw what happens when you add strings together or multiply strings by a number in a print command. What happens when you do the same to all the types above. What happens when you add different types together?

You define functions in Python by first using the keyword def. Python knows you are defining a function when this keyword is encountered. The **single line** to start a definition of a function consists of:

1. The keyword def
2. Followed by a space
3. Then a function name (select one that is meaningful)
4. Accompanied by an open parenthesis
5. Enclosing **arguments** used by the function (separated by commas)
6. Which end with a closing parenthesis
7. And a colon

Arguments are variables whose values will be used in the execution of the function. If there are no arguments, the parenthesis would be closed without anything inside. The actual instructions for the functions will be declared in indented lines below the line whose syntax is described above. Each line at the same level of hierarchy needs to be indented the same number of spaces. These tell the function which statements should be executed at which point. **If the indents are not correct, your Python notebook will complain!**

For example, you could have a function called **hello** that takes a message as an argument, with the instruction to print the message to the screen only if the message is longer than 10 characters. When the message is less than 10 characters long, the function can be told to print "The message is too short!". The code is presented below as an image. You might need to review [Section 4 of the Kaggle tutorial](https://www.kaggle.com/code/alexisbcook/conditions-and-conditional-statements) to fully comprehend the lines involving **if** and **else**. Type it into your notebook to see it work. Change the message to your own to see how it responds to different lengths.

Example of hello function.

def hello(message):
    if len(message) >10:
        print(message)
    else:
        print('The message is too short!')

message="Hello"
hello(message)

### Task 1.3 Group Challenge (easier): See if you can create the following function yourself.

Create a function in your notebook that takes a list as an argument, checks that it is at least length 10, and returns a list that excludes the first and last item in the list. For this challenge you will have had to completed [Section 5 of the Kaggle intro to programming](https://www.kaggle.com/code/alexisbcook/intro-to-lists). If you are having trouble, discuss it with your peer group!

### Task 1.4 OptionalGroup Challenge (harder): See if you can produce a Python code that:

Uses the pretrained image classification model VGG19 to see how it describes/categorises your chosen image. You can find [a long list of well-known pretrained models](https://keras.io/api/applications/) on the Kera website should you wish to try another pretrained model later. To complete this challenge, you need to:

1. Import keras.
2. Import your model
   1. To use the model VGG16, you would need to import VGG16 from the keras.applications.vgg16 library (i.e. from keras.applications.vgg16 import VGG16). You’ve seen this kind of code before in course (A).
   2. The way you import will be different depending on the model, so stick with VGG16 for now.
3. Import your model’s tool for decoding your model’s prediction to produce a human readable form.
   1. As this week’s lecture emphasised, your computer can only understand numbers so the prediction of the model will output numbers instead of classes and probability of each class being the correct one.
   2. To make this human understandable, you need to **decode** it. You need to import decode\_predictions from the library keras.applications.vgg16.
4. Finally, predict the class of your image with your chosen model.
   1. **Step 1:** use the pretrained classifier to classify your image:
      1. Assign a variable image\_path with the location of your chosen image file.
      2. Instantiate a model to use the weights trained on the ImageNet:

model = VGG16(weights=``imagenet”)

* + 1. Load your image specifying the right size, transform it to an array with the right shape (i.e. the size that the model expects). Here the size is 224 by 224 because that is what VGG16 expects. For InceptionV3 the size may be different:

img = keras.utils.load\_img(img\_path, target\_size=(224, 224))

image = keras.utils.img\_to\_array(img)

image = np.expand\_dims(image, axis=0)

* + 1. Predict categories with the function model.predict(image).
  1. **Step 2:** decode your computer’s output.
     1. Decode your prediction using decode\_prediction.
  2. **Step 3:** print out your model’s predictions.
     1. The decoded results output a list of 3-tuples where each tuple looks like (i, c, p). The c would be the class and p the probability that the model thinks the image belongs to c.
     2. You might need to extend your understanding of lists, tuples to fully understand this part, by reviewing the [Kaggle Python tutorial lesson on lists](https://www.kaggle.com/code/colinmorris/lists). This will help you figure out how to print the top five results.

1. Experiment with your own images and possibly other [models from the Keras website](https://keras.io/api/applications/) to print out the top five guesses of your chosen model.

## 2. Going Deeper with Pretrained Models – Deep Dream

### Task 2.1: What are Generative AI Models? – First Look

We already discussed generative AI models and how they might be considered to differ from predictive models that we discussed in course (A). Generative AI can take many forms. As we discussed in class, predictive AI builds a model that analyses data to produce a prediction (e.g. handwritten digit recognition; average housing price estimation). On the other hand, generative AI takes a category and/or a prediction and produces new data that would fall into that category or lead to that prediction.

Categories/

Predictions

Data

Model

The trick with generative AI, in particular, is to not produce data you already saw in the training data, but produce new data that would be recognised as belonging to the same category or related to the original data in some way.

**The main concept** **for generative AI**

The main principle underlying generative AI models lie in modelling the distribution of data so that data points belonging to the same category lie close to each other - you can think of the clustering algorithm we briefly discussed in course (A). If you sample data from that distribution, close to known items in a category, you can get completely new items in the category.

**Autoencoders: fundamental backbone**

Woking in high dimensions is hard – it makes computation really expensive (i.e. it takes a long time – maybe not possible at all). You experienced already how long it takes to train models in course (A). Autoencoders help with this by using an approach to **encode** data, mapping it to a lower dimensional space (called a **latent space** - e.g. taking something that has height, width and depth to be represented by height and width only – think of the biscuit tin examples from Week 2 lecture). Generative AI hinges on developing a clever way to **decode** the lower dimensional representation (e.g. height and width) to recreate the higher dimensional representation (e.g. the biscuit tin). The technique can be used for image compression and feature extraction, and inspired approaches in generating images, as the decoder can be used separately to **sample data points from the latent space** to create new content. In particular, it is the basis for the Variational Autoencoder (VAE) which we will talk about next.

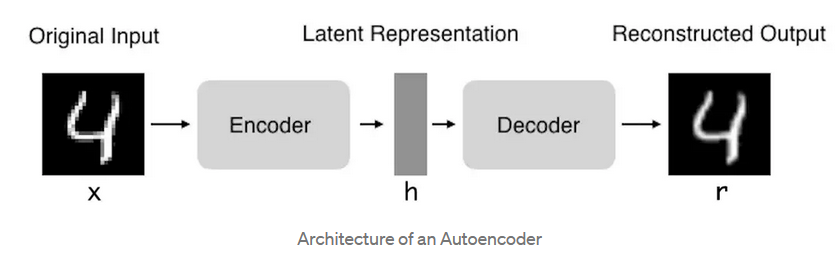


Figure 1. Copy right Nathan Hubens https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

**Variational Autoencoder (VAE)**

Variational Autoencoders were discovered simultaneously by Kingma and Welling in December 2013, and Rezende, Mohamed, and Wiestra in January 2014. (Francois Chollet, 2021 – course reading).

Variational autoencoders are defined by:

* A generative model adding probabilistic elements to the encoder and the decoder of an autoencoder. Probabilistic encoding of input data - not deterministic! The network learns a mean and a variance value (used to define a distribution).
* A loss function is set up to encourage the encoder to learn a prior distribution, while also trying to reconstruct the input data x. This means, for example, that it tries to reflect features of the original content but not the same.
* Points close to each other in the latent space yield similar images. Enables face swapping, turning a frown into a smile, animations by walking through the latent space, continuous morphing. Can be Strength over GAN.

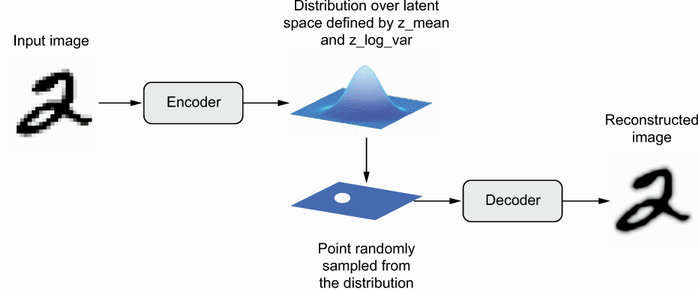


Figure 2 From https://arxiv.org/abs/1312.6114

### Task 2.2: Reflection on Using Pretrained Models and Explainability

One of the social concerns about AI currently is the lack of explainability of the model. This has been a concern already from earlier days before the proliferation of deep learning. Many researchers avoided neural networks for this very reason – simply put, they are a bit of a "black box", unlike models such as Decision Trees. This was not a big obstacle when AI algorithms were rarely incorporated into decision making processes in the real world, but as AI algorithms become more entwined with real world processes that can directly impact safety and ethics, we have come to demand more transparency. For example, if you are refused a loan at a bank, you would like to know why. An AI making loan decisions would struggle to provide coherent reasons for the everyday person.

* West, S.M., Whittaker, M. and Crawford, K. (2019). Discriminating Systems: Gender, Race and Power in AI. AI Now Institute. <https://ainowinstitute.org/discriminatingsystems.html>

In convolutional neural networks (we used this type of neural network in the lab on recognising handwritten digit detection in course (A)), there have been some interesting approaches for getting into the mind of the machine. One of these involve visualising the patterns captured/learned by the filters in the layers. In fact, in course (A) we explored some of these patterns using Tensorflow Playground to look at the architecture and patterns detected by each node in the network. We can extract these patterns learned by neural networks at selected layers then accentuate them to see what they look like.

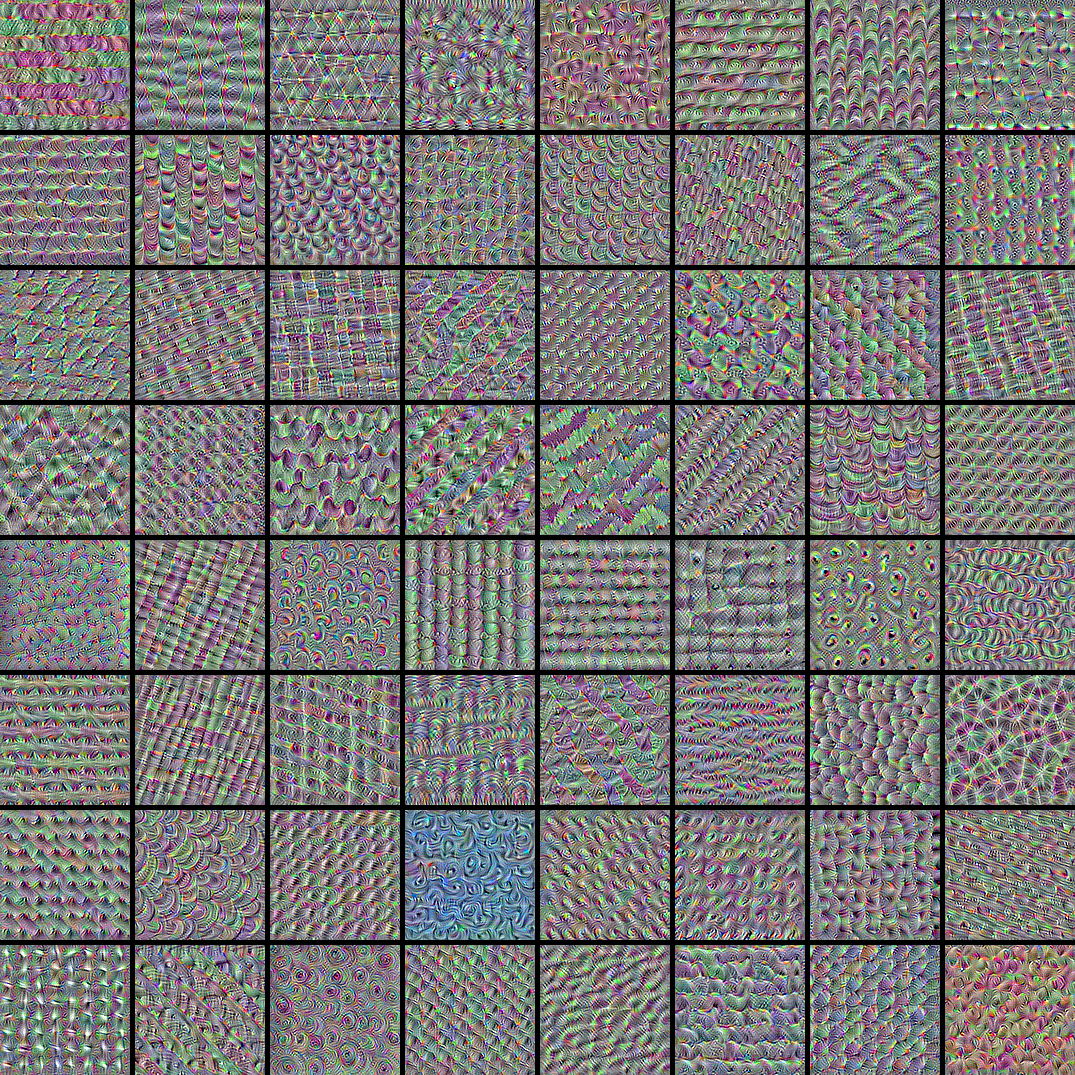


Figure 3. [Image created by fchollet to visualise patterns in different layers of VGG16.](https://keras.io/examples/vision/visualizing_what_convnets_learn/)

In 2015, this approach to visualising patterns led to one of the widest shared trippy images generated by deep neural networks, sparking what we now call Deep Dream (to be explored further in the next part of this book). You can learn more about how these patterns might be interpreted in this old Keras blog post on “[How convolutional neural networks see the world](https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html)”. You can also explore more visualisations here: <https://microscope.openai.com/models>.

There are many applications of pretrained models for transfer learning and generative AI, we will explore two in this section of the work book:

* Deep Dream: first released by Google in 2015 (Mondvintsev, Olah and Tyka 2015). Pareidolia artifacts imposes meaningful interpretation where there may be none (faces, eyes)
* Neural Style Transfer: introduced by Gatys, Ecker, and Bethge (2015). Applies the style of a reference image to a target image while conserving the content of the target image.

**Deep Dream**

Technique developed by the Google Deep Mind Team for reconstructing and visualizing selected layers of a trained deep neural network to be overlaid on the original image to give a dream-like or psychedelic effect. The process:

* Step 1: Loads a target image.
* Step 2: Defines a number of processing scales (called “octaves”), and resizes the original image to these scales.
* Step 3: Select some layers in your pretrained model to use to extract features from your image at these scales.
* Step 4: For every scale, starting with the smallest, run **gradient ascent,** to accentuate the extracted features andupscale the shrunk image to the next scale.
* Step 5: Re-inject the detail that was lost at upscaling time.
* Step 6: Stop when you get to the original Image size.

To reinject detail: take the original image, shrink it down, upscale it, and compare the result to the (resized) original image.

Read the [blogpost on Gary’s Notebook](https://www.garysnotebook.com/20190826_1) for some more examples of Deep Dream. Although this technique was popular earlier for images, there is no reason it cannot be applied to other data such as music, speech, and text.

**Neural Style Transfer**

Style transfer consists of generating an image with the same "content" as one base image, but with the "style" of another image. See Chollet 2020 in the reading list for more details. Developed by Leon A. Gatys et al 2015 - "[A Neural Algorithm of Artistic Style](https://arxiv.org/abs/1508.06576).

The algorithm works with the pretrained model to extract the features of an image at layers selected to represent either content or style. The algorithm uses the following loss functions to accentuate content from one image and the style of another.

* “content loss” (difference between the features of the base image and the features of the generated image)
* “style loss” (sum of the distances between the representations of the base image and the style reference image, extracted from different layers of a pre-trained model)
* “total variation loss” (reflecting local continuity to give the picture coherence)

Try out the demo and read the explanation at <https://reiinakano.com/arbitrary-image-stylization-tfjs/>.

### Task 2.3: Performing Code Archaeology

Before we jump into exploring the code for Deep Dream and Neural Style Transfer, this section introduces you to a few strategies for reading code to build on what you learned about presenting code to the Arts and Humanities in course (A).

Here, we want you to get into the mindset of an archaeologist digging for artifacts. One of the first things to do in archaeology when coming across a new find, is to recognise and classify the object. This is not that different for reading code.

* **Step 1 (Scan your code for familiar keywords):** so far in course (A) and in this course, we reviewed some basic concepts in programming. Try to highlight points in the code where these types of commands are found, annotate it with a comment, and where you need more information, link to the resources where more information can be found if necessary. The resources are useful for referencing later.
* **Step 2 (Use AI for better understanding):** use AI conversational agents (e.g. [Hugging Chat](https://huggingface.co/chat/), [Microsoft Copilot](https://copilot.microsoft.com/), ChatGPT – your GUID gives you access to Copilot) to get explanations of the code on a high level. Use snippets from the code and from the output (if there is an error) as part of your prompt to ask questions about the code and errors.  Note that you can get different responses from the AI depending on your prompt:
  + Break down the code into meaningful snippets to include in your prompt. Feel free to ask the AI about the role of specific lines in relation to the code block and vice versa as well as relationships between blocks.
  + Provide details about your expectations. For example, include in your prompt:
    - The purpose for which you are asking the question.
    - Your level of expertise in coding and in AI: e.g. "complete beginner in AI but some familiarity in Python".
    - Who the response should be for: e.g. a general audience, academic audience or for a specific educational background.
    - Requirements for appropriateness and relevance for audience: e.g. age range of audience. geographical region.
  + Make your request as specific as possible: e.g. instead of general requests to "explain the code to me", you might say "why is this line necessary for generating the image?".
  + Discuss your exploration with your peers. Find out what prompts they used and what they were able to find out. Compare with reliable information online and explanations in the course reading to make sure the AI is not making things up.
* **Step 3 (Reflection):** add markdown cells in your notebook to summarise and reflect on the responses and to add deeper explanations of the parts you don't understand.
* **Step 4 (Concluding the expedition):**

Consider:

* How you might change the code to apply Deep Dream to:
  + - Your own image
    - Use a different model
    - Use different parameters and layers (get the help of the AI to debate how these affect the outcome of the code).
  + How can you work with AI tools to generate modified code. And how would you test the code?
  + Ways in which you can use or creatively extend the code for your project?

### Task 2.4: Explore the code for Deep Dream

Perform code archaeology on [F. Chollet’s Deep Dream implementation](https://keras.io/examples/generative/deep_dream/) using the pretrained image classification model InceptionV3 which was trained in [ImageNet Dataset](https://image-net.org/). For your convenience, I have started the work to get you started and have included the notebook as [ipynb](https://moodle.gla.ac.uk/mod/resource/view.php?id=4099469) and [html](https://moodle.gla.ac.uk/mod/resource/view.php?id=4099470) files on the Moodle.

### Task 2.5: Explore the code for Neural Style Transfer

Perform code archaeology on [F. Chollet’s Neural Stye Transfer implementation](https://keras.io/examples/generative/neural_style_transfer/) using the pretrained image classification model VGG19 which was trained on the [ImageNet Dataset](https://keras.io/examples/generative/neural_style_transfer/). This code is a bit more complex than Deep Dream so go slowly – do not try to master the code in one sitting.

## 3. Transfer Learning with Pretrained Models

* Transferring knowledge from one task to another is called transfer learning. For example, “if you have already trained a model to recognize faces in pictures and you now want to train a new neural network to recognize hairstyles, you can kickstart the training by reusing the lower layers of the first network” (Aurélien Géron 2019 – course reading) to initialise your weights and biases.
* This is especially useful where you do not have a large amount data from the new task domain. In some cases error reduction of 25% is reported after using transfer learning.
* Since you are using layers from a previously trained network, transfer learning is often tied to a selection of pretrained models. In fact, it is good practice to look for pre-trained models in your area of interest, even if you do not end up using them.
* Transfer learning and pretrained models help speed up training because you do not need to start from scratch.

# Exploring Generative Models – Part II (Week 5 - 7) – to be available shortly.

In this section we will explore some more generative models in AI. In the section “Exploring Generative Models – PART I”, you already learned about using pretrained models to apply Deep Dream and Neural Style Transfer. The former accentuates features of an image recognised by a pretrained model to give a dream-like effect. The latter tries to associate different layers of a pretrained model to style and content, to create a new image with content from one image and style from another.

In this section we will selectively dig deeper into three of the foundational models of generative AI: generative adversarial networks, transformers, and stable diffusion. The terminologies used in these models can be inscrutable at the beginning – but don’t worry! You will start to get the idea as you talk about them in your peer group discussions.

## 1. Understanding the algorithms

So far, in the lab, we’ve explored the concept of Autoencoders and Variational Autoencoders and how these relate to Generative AI. However, mainstream applications that have been talked about in the news such as that for image generation (e.g. DALL-E 2 & 3), text generation (e.g. ChatGPT, Hugging Chat, Glasgow Chat, Microsoft Copilot), video generation (e.g. Stable Video Diffusion) and code completion have one of the following as a building block in their architecture: **generative adversarial networks**, **transformers**, and **stable diffusion**. There are undoubtedly a large number of other techniques at play and variation but understanding these three algorithms will help you work towards mastery of generative AI in the longer term.

### **Task 1.1. Understanding the Architecture: Generative Adversarial Networks (GANs)**

GANs consist of two neural networks: a **generator** and a **discriminator**. The generator creates new data samples, while the discriminator tries to distinguish between the generated samples and real (human generated) samples. The generator is trained to create samples that are indistinguishable from the real samples, while the discriminator is trained to correctly identify whether a sample is real or generated. To put it another way, the objective of the generator is to fool the discriminator while that of the discriminator is to unearth the deception. The competitive relationship is where the word “adversarial” in the algorithm’s name comes from. The algorithm has been shown to be reasonably effective regardless of where you place the focus of your application, discriminative task (e.g. detecting images of dogs) or generative (e.g. creating an image in the style of the artist [Frida Kahlo](https://en.wikipedia.org/wiki/Frida_Kahlo)), because it improves both models concurrently. The generative model has further been recently used to help understand structures of proteins and cells which can help detect diseases: e.g. [3D reconstruction of lung tumour](https://link.springer.com/article/10.1007/s11042-023-15232-0).

The algorithm was suggested by Goodfellow et al. in 2014 in their paper “[Generative Adversarial Networks](https://arxiv.org/abs/1406.2661)”. Although the algorithm has been used most frequently in the context where both the discriminator and generator are neural networks, there is no reason they need to be neural networks as the following overview of the GAN workflow demonstrates:

1. The generator creates new content.
2. The discriminator evaluates whether the content is real or generated.
3. The generator adjusts its parameters to create more realistic content after assessing the discriminator’s evaluation.
4. The discriminator adjusts its parameters to better distinguish between real and generated content.
5. Repeat steps 1-4 until the generator is able to create content that cannot be distinguished by the discriminator.

GANs have many generative AI applications, including image generation, text generation, and music generation, among others. Recall how we sketched out a diagram for machine learning workflow. How would you sketch out a diagram for a GAN?

If you are interested to learn more of the technical details, read:

* [Foster (2019) “Generative Deep Learning”](https://eleanor.lib.gla.ac.uk/record=b3896939) Chapter 4 (course reading)

### **Task 1.2. Understanding the Architecture: Transformers**

In AI for the Arts and Humanities (A), we looked at a neural network that **predicts** the next item in a sequence after a given sequence of items (e.g., Shakespeare text generation; Bach chorale generation). The code provided was a special type of **recurrent neural network**. The algorithms received, for example, the input sequence of characters to output another sequence of characters (including space and punctuation): e.g.,

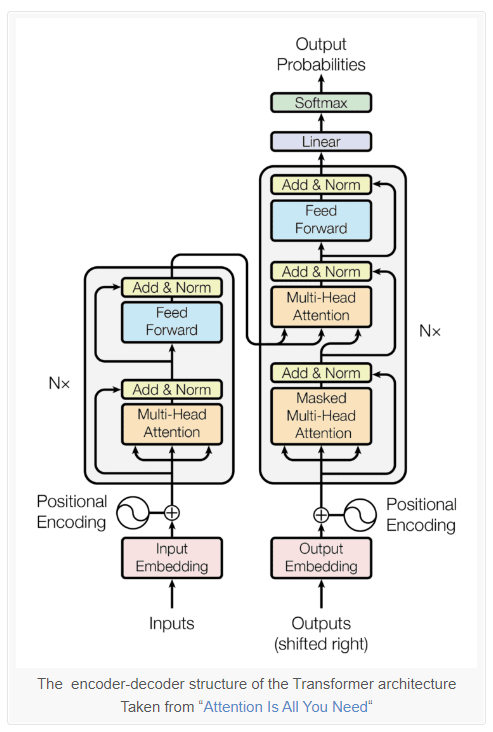
“To be or not to b” 🡪 “o be or not to be”

“o be or not to be” 🡪 “ be or not to be ”

“be or not to be ” 🡪 “be or not to be t”

…

This kind of model is called a sequence-to-sequence model. General sequence-to-sequence models take sequences of variable lengths to output sequences of variable lengths. In the Shakespeare generation task, the input and output languages were the same – Shakespearean English. In a cross-language translation task, the English sentence “Let’s go” can be translated into the one word Spanish sentence “Vamos”. In such an algorithm, each word is **encoded** as a sequence of numbers (called a context vector) by a neural network. Although we did not do this for the Shakespearean text generation task, for a general sequence-to-sequence model, this encoding will be passed on to a second neural network called a decoder. The decoder will process the input to output the probability of each word (in the target vocabulary – e.g. Spanish) being the next word.



Transformer architectures are examples of a sequence-to-sequence models. It was developed by Vaswani et al. in 2017 in the paper “[Attention is all you need](https://arxiv.org/abs/1706.03762)”. Transformer also employs a encoder-decoder structure. Unlike previous sequence-to-sequence models, however, the transformer architecture employs components called **self-attention mechanisms** to weigh the value of different parts of the input sequence. There could many of these to form **multi-head attention** that capture different contexts of the language. Transformers also take into account the **positional information as part of the encoding**, distinguishing the two phrases:

“put sugar in the coffee” **and** “put coffee in the sugar”

We will not go into details about how attention is implemented. However, if you are interested in the technical details, you can have a look at the articles and videos below:

1. [Foster (2019) “Generative Deep Learning”](https://eleanor.lib.gla.ac.uk/record=b3896939) Chapter 9 (course reading)
2. [The Transformer Model (Machine Learning Mastery)](https://machinelearningmastery.com/the-transformer-model/)
3. Stat Quest videos presented by Josh Starmer:
4. [Sequence-to-sequence encoder-decoder model](https://youtu.be/L8HKweZIOmg)
5. [Attention for neural networks](https://youtu.be/PSs6nxngL6k)
6. [Transformers](https://youtu.be/zxQyTK8quyY)

### **Task 1.3. Understanding the Architecture: Stable Diffusion**

Stable Diffusion is a type of artificial intelligence model designed for generating and manipulating images, recently, most famous for text-to-image generation (e.g. [Stable Diffusion Online](https://stablediffusionweb.com/)). Imagine it as an artist that can create a wide range of visuals, from paintings to photorealistic images, simply by interpreting the words you provide.

At its heart, Stable Diffusion operates by:

1. Understanding the complex relationship between text descriptions and the vast world of visual imagery
2. Visual imagery compressed to essential information which is mapped to what we call a **latent space**.

It is like having an extensive visual library in its memory and the ability to understand language, enabling it to translate textual prompts into detailed images.

We encountered the term **latent space** before in relation to autoencoders and variational autoencoders. You can think of latent space as a vast, multidimensional canvas where every possible image can be mapped. This space will usually be lower in dimension than the original image which can be represented by millions of pixels. Nevertheless the latent space is intended to have enough dimensions to capture the essential characteristics of a given image. Some of the points on this latent space would correspond to images we have seen before (for example, those from the training data). Others represent new images we have not seen before which can be reconstructed to generate new images.

**How is Latent Space Used in Image Generation?**

**Search**

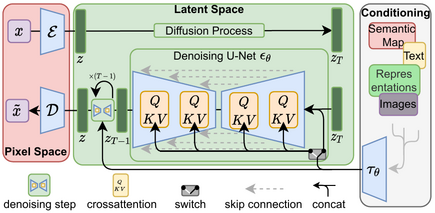
When generating an image, a model like Stable Diffusion navigates this latent space to find a point that corresponds to the textual description provided. This process involves transforming the text into a query (represented numerically) that the model can understand, which then guides the search through latent space to find a matching image representation.

**Reconstruction**

Once a specific point in latent space is identified, the model "decompresses" this point back into an image. This involves translating the condensed information captured in the latent space (remember that we only keep the essential information in the latent space) back into a full-fledged image with all the necessary details and nuances.

The "diffusion" part of Stable Diffusion refers to a specific technique used to generate images, going from random patterns of pixels (or noise), **denoising** until randomness is replaced with structured details. It's akin to starting with a picture out of focus (i.e. you can only see splatters of colours and shapes) to a picture in focus, or like tuning a radio going from static to recognisable voices/singing.

At training, you add noise to known images when to train the diffusion process. Then you apply the diffusion process to sampled points in the latent space, before reconstructing the image.



**Related Images in Proximity**

For tasks like image editing or style transfer, the model moves or adjusts the position within the latent space relative to an original image (the image you are editing or want the tyle of). This allows for the modification of certain features while retaining others, based on the textual prompts or commands.

In essence, Stable Diffusion application to text-to-image acts as a bridge between the written word and visual art, enabling the creation of images that were previously only possible in the imagination. Its blend of technology and artistry offers a new realm of possibilities for creative expression and visual storytelling.

If you are interested in more technical detail, check out the following sources:

1. Stable Diffusion - Wikipedia. <https://en.wikipedia.org/wiki/Stable_Diffusion>
2. Stable Diffusion - Scholars at Harvard. <https://scholar.harvard.edu/files/binxuw/files/stable_diffusion_a_tutorial.pdf>
3. Understanding Stable Diffusion from "Scratch" | Binxu Wang. <https://scholar.harvard.edu/binxuw/classes/machine-learning-scratch/materials/stable-diffusion-scratch>

## 2. Perform Code Archaeology on existing code

We have provided links to example codes below for specific types of GAN, Transformers, and Stable Diffusion. There are many more out there (e.g. search for the keywords on Hugging Face – distinct from Hugging Chat.)

* GAN example code: [Conditional GAN](https://keras.io/examples/generative/conditional_gan/)
* Transformer example code: [Text generation with a miniature GPT](https://keras.io/examples/generative/text_generation_with_miniature_gpt/)
* Stable Diffusion example code: [A walk through latent space with stable diffusion](https://keras.io/examples/generative/random_walks_with_stable_diffusion/)

The pages linked above provide links to a keras.io website on GitHub and/or Google Colab notebooks. The links for the GitHub might take you to a python file which you can import to Jupyter Notebook, but you can also search the website directory for notebook files as well. These can be useful for digging through the code.

### Task 2.1. Perform ‘code archaeology’ (we defined this in the exercises for Weeks 3-4) on at least one of existing codes above. Attempt to get a high-level understanding of the code.

That is:

1. Try to sort out familiar terms in the code (e.g. for printing, importing libraries and the names of libraries, assigning variable) and those that are not familiar.
2. Employ your favourite AI friend (e.g. Hugging Chat, Microsoft Copilot) to find out more about those terms that are unfamiliar.
3. Try to identify bits that align with the steps in the workflow described in the section **1. Understanding the Algorithms** – use the Navigation Panel for this document to jump between this and that section. To make the navigation panel visible, tick the box in the **view** menu.
4. Copy and paste code into your favourite conversational AI and engineer a prompt (as per instructions in **Task 2.3 of Exploring Generative Models – Part I**) to understand complicated bits.
5. Discuss your findings and understanding with your peer group.

### Task 2.2. Consider the following points.

1. How might you change the code to use:
   1. Your own data.
   2. Test the effects of different parameters.
2. How can you work with the help of your favourite conversational AI (e.g. Microsoft Copilot) to generate modified code. How would you test the code to know that it is doing what you think it is doing?
3. Ways in which you can use or creatively extend the code for your project?

## 3. Some more ready-made tools to explore

So far, in this course, we’ve explored some generative AI application (Week 2) that require no coding, and, some pre-trained models (Week 3) that can be combined with your own Python code, as well as some existing code to start from scratch in training a model. In addition to these, however, there have been a number of initiative to democratise AI for general access, allowing training without coding, and/or, the use of language models from your own computer.

Here we will explore two resources which are especially of interest.

### Task 3.1 Introduction to llamafile

If you do not know already, OpenAI’s large language model GPT-4 **is not open source** (i.e. we have no access to the training data or code under the hood used to create the model). We only have access to the use of the final model through applications such as ChatGPT, or, through OpenAI’s API which allows you to interact with the large language model at a cost. In contrast, [Meta](https://www.meta.com/gb/) took the bold move to make their large language model [Llama 2](https://llama.meta.com/llama2) and data open source free of charge.

Furthermore, in November 2023, [Stephen Hood announced](https://hacks.mozilla.org/2023/11/introducing-llamafile/) the launch of [llamafile](https://github.com/Mozilla-Ocho/llamafile), which is a single file that can be *executed* to let you run your own large language model LLAMA 2 on your own computer. The model can be run through your browser. This in turn allows you to embed into your Python code. You can “borrow” the use of openai python package (you should have installed this in your Anaconda environment earlier this semester!) to talk to interact with the large language model.

Follow the instructions at [Mozilla-Ocho’s GitHub llamafile repository](https://github.com/Mozilla-Ocho/llamafile) to get the llamafile to work on your computer. You need to download the file and make the file into an executable file. The instructions for this are [on the GitHub page](https://github.com/Mozilla-Ocho/llamafile), and the steps are different for different operating systems. Note, that **for Windows, the maximum permitted llamafile size is 4GB.**

Once you have managed to start llamafile, this creates a local server for the language model at <http://localhost:8080/v1> (note this is not accessible to others on the Internet – just those using your computer!). This means that you should be able to talk to the language model from your Jupyter Notebook using the following code.

from openai import OpenAI

client = OpenAI(

base\_url="http://localhost:8080/v1", # "http://<Your api-server IP>:port"

api\_key = "sk-no-key-required"

)

completion = client.chat.completions.create(

model="LLaMA\_CPP",

messages=[

{"role": "system", "content": "You are ChatGPT, an AI assistant. Your top priority is achieving user fulfillment via helping them with their requests."},

{"role": "user", "content": "Write a limerick about python exceptions"}

]

)

print(completion.choices[0].message)

**This code is not doing anything with OpenAI**! It is just using the python library to talk to the large language model hosted on your computer. The user request in this example was **"Write a limerick about python exceptions".**

Although you are not using OpenAI APIs, [Open AI’s documentation on how to create chat completions](https://platform.openai.com/docs/api-reference/chat/create) can be useful to help you customise your request to the large language model.

### Task 3.2. Introduction to Teachable Machines

Apart from llamafile, Google has also been maintaining [Teachable Machines](https://teachablemachine.withgoogle.com/) since 2017. This might require a Google account but allows you to use your own data to train models to classify images, audio and body positions. With this, you merely need to upload your content split into training and test and the Teachable Machine will do the rest.

Tools such as llamafile and Teachable Machines, allow you to explore the intermediate space where you do not need to code everything from scratch but can embed models and your own data into your projects.

## 4. Developing your project idea

As mentioned earlier in this course, **you do not need to code anything from scratch** to complete the project for AI for the Arts and Humanities (B). However, **you do need to talk about the tools you decide to use in a informed way** to align with the discussion about machine learning and AI algorithms, training/test data, and how the data is processed through the workflow, as discussed in course (A) and (B). If your tool is proprietary and this information is limited, you would still be expected to say that explicitly and discuss what you have been able to find out.

Regardless of whether you code or not, **you will be expected to present your project as a Jupyter Notebook** on your GitHub repository – see the [Assessment Submission Guide on the Moodle](https://moodle.gla.ac.uk/mod/page/view.php?id=4099443). While it is not a requirement, the Python you learned in course (A) and (B), and the code examples in this work book can facilitate explaining the algorithm and/or workflow of your tools and/or project. For example, **you could include code snippets** as one way of engaging your arts and humanities with these aspects.

### Warm-up Task:

Discuss [the submission guide on the Moodle](https://moodle.gla.ac.uk/mod/page/view.php?id=4099443) with your peer group. Questions to review with your group:

1. What are the frequently asked questions about the assessment in your group, for example, about the project criteria? Collate these and submit unresolved questions on the forum.
2. Brainstorm to propose different types of output that you might present in your notebook portfolio to engage your audience with your chosen object. Come up with options in your group. For example, it could be audio, visual, or textual work (e.g. stories, art), scholarly presentation, and/or even something fun like a game. The options are limitless.
3. Remember, you only have a few weeks and limited resources - keep the scope small and simple to ensure quality and originality rather than going for complexity. Discuss with your group: how would you make sure that your project scope is feasible?

One of the criteria for the project is that your project **must start with a collection object and/or material**. See the list in [Week 4 section of the Moodle](https://moodle.gla.ac.uk/course/view.php?id=41137#section-8) and the [presentation from Bob McLean](https://uofglasgow.zoom.us/rec/share/A2I7PwiNBXwZgdZbDrWFnCvVQ0xZJQ2rxJQbFQDu3crU35kjFwZd06By0zL06dBn.5qFcxKfaSShtXiTi?startTime=1706613292000&pwd=ePuaJz7thiEkX4n5Vp0edrDICYLXxKsq) for details on this year's options. Once you have chosen an object, the aim of your project is to employ “Human-AI collaboration and creativity to engage a general audience with cultural heritage”. You are expected to use "generative AI" (interpreted broadly) to identify ways to engage the public with the selected object.

You might think about ways to raise a new perspective or question about your chosen object. For example, you could consider relating the object or your output to a current social interest in an unexpected way.

To design your project you will have to engage with three main activities as part of your creative process: **selecting and digitising data**, **generating new data**, and **combining data** as a holistic output to present the outcome. This is not a linear process, or even a cycle, but an interactive process between each of these components.

### What is data?

All AI projects, whether discriminative or generative starts with gathering data. What you gather, and how you gather data, is an integral part of your creative process.

In principle, anything can be data. You already saw examples in course (A) and this course such as images, text, audio, string, numbers, and boolean values (True/False) that has been used as data**. It isn't what it looks like that defines it as data.** Rather it is how it is used, analysed and interpreted that determines its nature, either as an every day object or as data.

For example, remember how the photo of a hedgehog could be perceived with your eyes as a picture of a cute animal which might even elicit an emotional response. It was just as easily read by the computer as an array of numbers, ready to be transformed in various ways (e.g. like in Deep Dream). In machine learning (or AI), data is often represented as something called a **tensor** which is similar to that array that you saw. AI algorithms transform and manipulate tensors to output new tensors that generate or discriminate content. This is where the name of the package tensorflow comes from.

### Digital data

One thing that is clear is that, for a computer to use, analyse, and/or interpret object as data, in the current system, the object needs to be transformed into a digital object. This process is called something different for different disciplines and tasks (e.g. digitisation; analogue to digital conversion; encoding). For objects like pages in a medieval manuscript, it might require 2D digitisation. For objects like sculptures or buildings, it might require 3D modelling. In many cases it might merely be a question of **taking a digital photo or video** with your phone. In other cases you might need to go further to process the digital objects for optical character recognition (OCR) such as [Transkribus](https://readcoop.eu/transkribus/) or [tesseract](https://github.com/tesseract-ocr/tesseract) to extract digital text from the image or PDF, and/or annotate specific parts of the objects.

### Collection objects as data

As highlighted earlier, you will be selecting objects either from the [Special Collections of the University](https://www.gla.ac.uk/myglasgow/archivespecialcollections/). In the case of the Special Collections, the objects presented in the Lecture of [Week 4](https://moodle.gla.ac.uk/course/view.php?id=31647#section-5) are not yet represented in digital form, except for the travel journal that Bob transcribed. Therefore you will need to start by choosing a **method for transforming the object into something digital**.

### Task 4.1. Establishing the context

Review the collections either by reviewing the presentation in Week 4 by the Special Collections. Choose a couple of objects that inspire you. Try to find information on the following questions:

* What is the object? In particular, can you find out any information about its origin, history or use?
* What is the value of the object? What distinguishes it, why is it interesting, and/or who would it be of interest to?
* Are there other known objects that are similar or are related to the object by context, history or origin?

These questions help you to provide clarity of context (first criteria for assessment). It also helps you to think about what new perspective can you bring to the object.

**Ask yourself:** can you use the object to bring to light a question about groups currently underrepresented in collections? Does the object have an unexpected connection to another object/material/information of current cultural, social or technological interest? Perhaps something of current interest in the news?

### Generating New Data

In [Week 2](https://moodle.gla.ac.uk/course/view.php?id=41137#section-6) you used some Generative AI tools to create content relevant to, for example, a Wikipedia article. You also explored some pre-trained models and existing code as well as llamafile and teachable machines. Regardless of whether you use tools, code it yourself, or explore some intermediate approach, think about what kind of AI generated content might engage your arts and humanities audience with your chosen object.

**If you did not carry out the reflection included in the lab of that week, this is a good time to review those sections**.

If you like AI art, you might want to explore featured AI artists at [AIArtists.org](https://aiartists.org/ai-artist-founding-members). If you like games, you might also want to explore [AI Dungeon](https://play.aidungeon.io/) (warning: this tool has many blind spots, bugs, and limitations). You can also code with your AI assistance (remember to check that it actually works!). Creativity is not limited to creating art, games, and design, but extends to reaching new potentials in our everyday creative practice to engage in novel explorations, connections, transformations, and therefore can be even applied to programming, academic practice, science and scholarly presentations in equal measure.

### Content generated by you

Think about what content you can generate to complement the content generated by AI. This can be images, text, and/or audio, but it can also be in the form of decisions, annotations, organisations, selection and curation of content as well as interpretation. Think about how you and AI can help each other to discover new things you would not have learned on your own to enhance the project.

### Task 4.2. Think about AI-Human Interaction

Reflect on the relationship between AI generated content and content generated by you. You might want to think of them as informing each other in a feedback loop, or complementing each by carrying out different parts with a shared end-goal, rather than independent and/or discrete activities.

Draw a diagram of the process and then compare with your group. This does not have to be detailed. This is just some initial talking points for investigating further with the group, the class and for your presentation to be submitted in Week 7.

### Combining Data

### Reflecting on your plan

How do the generated data and original data created from the object come together to meet the objectives for which you selected the collection object? Reflect on the Lovelace-Boden-Ridler Test discussed in Week 1. Also, reflect on the full criteria for the evaluation report on the Moodle. How would you evaluate the proposed output.

### Task 3.1. Planning Evaluation

Reflect with your group what steps might be involved in evaluating your project. At this point you are not expected to have a full plan, just some ideas to start you and your group thinking.

# Upcoming: Guest lecture and self-managed project development (Week 8 onwards)

Guest lectures:

* Week 8: AI and Art with Marion Carre
* Week 9: AI and Music with Tim Duguid
* Week 10: AI ethics (TBC)
* Week 11: Project Q&A

Labs will be your own project work with the support of tutors and lecturers.