Lab assignment 1: Image recognition using deep networks. Learning-based exercises

By Ben Harvey

In this lab assignment, you will explore questions of visual image processing. The assignment is divided into two parts.

Learning-based exercises (10% of your course grade) are focussed on improving your understanding of visual image processing and the methods used to investigate this.
- These are best completed in class, with considerable feedback from your teachers.
- Your teachers will grade your work as you go and suggest ways to improve your work.
- Your grades can increase if you make these improvements. If you make these improvements, show them to the same teacher who gave you feedback on your original answers.
- We suggest that each pair work together with one other pair during these exercises. These two pairs can explore the exercises together and help each other, which improves learning.
- You can show your answers to your teacher together, which is less work for them.
- However, each pair should complete and submit their own final answers in a separate document so that we have a record of the work you were graded on.
- Therefore, it is possible for one pair to do more work separately to improve their answers more after feedback from a teacher. In that case, the two pairs can receive slightly different grades. If you do this, make clear to your teacher which pair contributed to the improvements.

- Exercise Three of this learning-based assignment asks you to study two topics for the mid-term exam that we will not cover in detail in the lectures. These are important topics that we find students understand best after self-study. Here we want to assess your own individual understanding, which is your own individual responsibility. Nevertheless, it may be helpful to study together with your teammates and get input from your teacher. Please note that the deadline for submitting the lab assignments is after the mid-term exam, so **Exercise Three effectively has a deadline at the mid-term exam**. Your answers will be given during the exam, so contribute to your exam grade.

The **Challenge-based exercise** (10% of your course grade) focusses on your teachers assessing your understanding of the content you have covered.
-These questions are found in a separate document.
- It is best to do these after you have done (most of) Learning-based Exercise One.
- This exercise can be completed during class or outside of class time.
- You will not get input from your teachers, except to clarify the questions.
- You should **NOT** work with another pair: each pair should work alone.
- Each pair should submit a separate answers document specifically for the challenge-based exercise, at the separate submission link on Brightspace.

You will code for the exercises in both assignments using the Keras library for Python. Keras is a high-level library for building artificial machine learning networks, specialising in deep learning networks. Python is used to address Keras, while Keras primarily calls the open-source TensorFlow library, developed by the Google Brain team. TensorFlow in turn is written in Python and C++. As a result, Python must be installed too, which is typically done automatically when installing Keras.

Keras is therefore best viewed as an interface for TensorFlow, while TensorFlow is the underlying machine learning framework. Keras gives a more intuitive set of functions, making it easy to develop machine learning models, particularly in the context of deep learning networks. Keras, like TensorFlow and Python, has excellent cross-platform support, and importantly makes it very easy to run models on their CPUs, GPUs or clusters of either. Because artificial learning networks rely on many simple computations running in parallel, using GPUs is ideal for this application.

Because we will use relatively simple networks in this class, it is feasible to run these exercises on your own computers. However, modern laptops vary considerably in their CPUs and GPUs. A faster GPU will considerably reduce your waiting times, so if you have a choice of computers, choose the one with the better graphics card. Or Google Colab will run your code on Google's servers, fast.

You should work in pairs on these exercises, though you are welcome to work together with one other pair on the learning-based exercises in this document. You should get help from a single teacher (who is working in your part of the classroom) during this learning-based assignment. To help you keep working with the same teacher, please sit in a similar place in each class. We expect you to work on these exercises in class time so you can work together and get help from your teacher. We expect you both to work on the exercises but submit a single version that you both agree on. **It is only acceptable to miss these classes when your partner agrees to synchronise your work after you have both completed it. You can only get help from your teacher during class time. If you won't be able to attend most of these lab classes, you should leave the course now or find a partner who is willing to work with you extensively outside of regular class times**. If you find that your partner is not contributing to your joint assignment, please inform your teacher to avoid this slowing you down and affecting your grades.

We suggest group members doing these exercises on your individual computers simultaneously. Most students choose to do this in Jupyter Notebook, a Python Integrated Development Environment (IDE). Completing these exercises individually improves (student) learning and also makes it easier to find mistakes. Don't rely on your partner's answers if you don't understand why they are correct: this is meant to be an interactive collaboration, so ask your partner to explain. An excellent way to decide together on a final version of your **code** is Google Colab, which allows you all to work on the same document together, interfaces nicely with Jupyter Notebook, and runs your code on Google's fast servers. If your group gets stuck on a question or you can't agree on an answer, ask for help from your teacher.

When you are both happy with your own answer, work together to finalise your agreed answer in a shared text document. Google Docs is an excellent platform for working together on a shared **text** document. At the end of the assignment, **export this document <u>as a PDF</u> and submit this on Brightspace**. Each group should submit a single document containing both your names and student numbers. You will both receive the same grade. You will be graded for the depth and completion of your answers.

All the words and code in this document should be written by you and your partner only. It will be compared to other groups' submissions and online sources to check for plagiarism (i.e. fraud). Similarly, if you share your work with other groups and they submit plagiarised parts of your assignment, this is considered fraud by both groups.

University examination regulations currently define fraud as 'the copying of pieces, thoughts, reasoning of others and passing them off as one's own work'. The term 'others' here traditionally implies people have generated the original text, but is currently interpreted by the exam committee to include automatic text generators such as ChatGPT (statements referring to ChatGPT below apply to all automatic text and code generators). As such, submitting automatically generated text or code as your own work is considered fraud. We try to ask questions that ChatGPT answers poorly. ChatGPT often gives plausible sounding but factually incorrect answers.

If you wish to include text or code written by other people or automatic text or code generators in your submission, it should be clearly labelled as a quote and attributed to the original author or text generation model. This is not fraud as you are not 'passing them off as one's own work', but our grading will take the source into account.

Many questions begin 'Discuss with your group, write, then describe to your teacher…'. In questions like this, it is generally best to start by asking every group member's opinion. Then work on a written answer together. Then explain your answer to your teacher. You can also ask your teacher to read what you wrote, but they will often ask questions. You may like to update this answer after talking with your teacher, but please tell your teacher what changes you made next time you talk.

Many questions build on previous questions being completed correctly, so you should be confident of your answer before using it in further questions: ask for help if you are unsure. The assignment often shows what a correct output looks like so you can check you are on the right track before moving on: if your output looks different, try again. But there are various points clearly marked (NEW TOPIC) where you do not rely on previous answers or code. If you get stuck and can't get help immediately, you can move on to the next topic until your teacher can help.

In the following text, explanation is plain text, instructions are **underlined**, questions to answer are labelled and numbered, and Python code snippets are written in `different font`. The number of points available for each questions gives an idea of the expected depth of your answers: try to make at least that many distinct points in your answer.

Installation:
If you do not already have Python installed, download it here and install it:
https://www.python.org/downloads/ (*Note: This assignment is targeted at Python version 3.8, but other 3.x versions should work as well*). Python comes preinstalled on most Linux distributions, and is available as a package on all others.

An IDE is not required for Python, but it is recommended for debugging. There are many different IDE's available. Optionally, install an IDE such as PyCharm:
https://www.jetbrains.com/pycharm/
Or Jupyter Notebook, which interacts with Google Colab: https://jupyter.org/install

To install Keras for Python, open your console or IDE and enter the following commands:

```
pip install --upgrade tensorflow
pip install keras
```

Python, unlike R and Matlab, does not come with inbuilt graphing functions. Therefore installing a graphing package such as matplotlib is recommended:

```
pip install matplotlib
```

Now load the Keras library in your code:

```
from tensorflow import keras
import numpy as np
```

And optionally, to simplify plotting codes:

```
import matplotlib.pyplot as plt
```

Exercise One: Identifying handwritten numbers

We will begin using a very simple image recognition example: classifying hand-written numbers. This is a very useful ability for computers as it allows mail carriers to read hand-written postal codes and house numbers, and thereby sort mail automatically. It also allows banks to read numbers from cheques.

Discuss with your group, write, then describe to your teacher, a list of at least 3 applications where automatic recognition of hand-written numbers would be useful. (**Question 1**, 2 points)

We will use a database of labelled handwritten numbers, called MNIST.
First, download MNIST and store both train and test tuples:

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

MNIST contains a training set of 60,000 grayscale images (each 28x28 pixels) of hand-written numbers (`x_train`) together with the numbers shown in each image, the labels (`y_train`). It also contains a similar test set of 10,000 images and labels (`x_test` and `y_test`). You can verify these numbers by printing the values of these variables by affing .shape to them, i.e. print(`x_train.shape)`.

Here we will use two types of artificial learning network. Before using a deep convolutional network, we will test a multilayer perceptron. This is a type of artificial neural network where all nodes in each layer are connected to all nodes in the next layer. Therefore, this is not a convolutional network because there is no spatially-restricted convolutional filter, and no use of spatial relationships.

Data preparation:
So we will start by flattening the two spatial dimensions to convert from a 60000x28x28 training set to a 60000x784 training set in which spatial relationships are removed.

Use the function `numpy.reshape` to convert the flatten training and test set images from 28x28 pixels to a column of 784 pixels for each image (row). Save the results as new variables called x_train and x_test to avoid overwriting the original images. These should have dimensions 60000x784 and 10000x784 respectively (use the call `.shape` on the variables to check this).

Now rescale x_train and x_test to values between zero and one by dividing each variable by 255.

The labels are currently specified as numbers between 1 and 10. For our network, these numbers must each be separate network units in the output layer. Use the function `keras.utils.to_categorical(var, 10)` to convert the train and testset labels to two new variables, called y_train and y_test. For each label, there should be 10 elements, 9 of them zeros and 1 of them a one.

Model definition:
You will want to keep a record of the code you write here so you can easily make modifications and run it again.
For our multi-layer perceptron (MLP), we will pass our 784 input units (flattened pixels) into a 256-unit fully connected hidden layer. This in turn feeds into our label (output) layer whose activation follows a softmax function to give the probability that each image is each digit. We initialise this model using the function `keras.Sequential()`. The fully-connected layers are defined by the Keras function `keras.layers.Dense()`.

So, to make the MLP model described above, enter the following code:

```
model = keras.Sequential()
model.add(keras.layers.Dense(256, input_shape=(784,)))
model.add(keras.layers.Dense(10, activation='softmax'))
```

To check the resulting model is what you expect, use:

```
model.summary()
```

If this looks OK, compile the model with suitable loss functions, optimisation procedures, and performance measures, as follows:

```
model.compile(loss='categorical_crossentropy',
optimizer=keras.optimizers.RMSprop(), metrics=['accuracy'])
```

Training and evaluation

Now we will fit the model to our training set, keeping a history of the performance at each stage. We will use 12 training epochs. We will set aside a random 20% of our data to check performance at each training epoch (which don't change between epochs). In each training epoch, we will use 128 image-label pairs per batch to improve computational efficiency. We will use the `verbose` argument to tell us what is happening in each epoch.

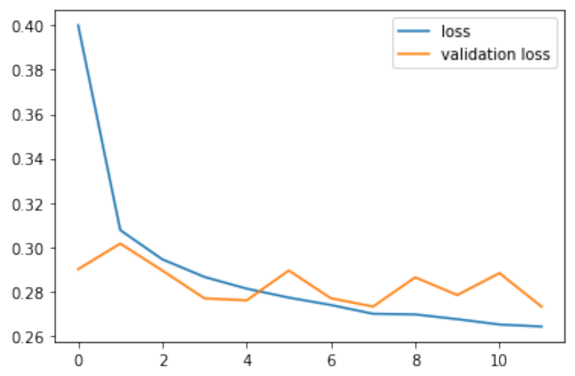
To fit the model as described above, enter the following code:

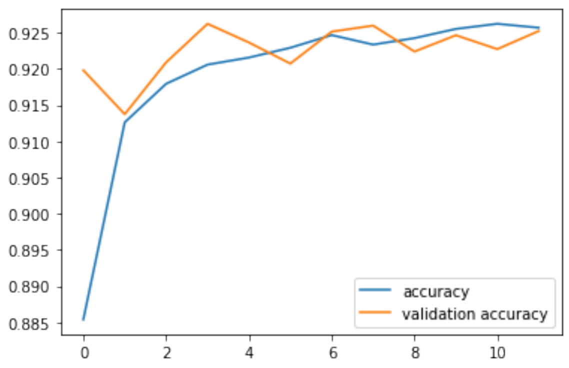
```
history = model.fit(x_train, y_train, batch_size=128,
epochs=12, verbose=1, validation_split=0.2)
```

For questions where you generate a plot, table or code, you should copy this in your answer document (maybe using a screenshot).

In the text from your console, you should see how long it took for each epoch to run and the training performance history. You should see that the time taken for each epoch doesn't change (much) between epochs,

From the output variable `history`, plot the training history loss and accuracy for both the training set and validation set. You will need to do this often in later questions. The result should look something like the images below. If you can't get a plot like that, ask for help here.





Write your answers to questions 2-3 (below) together before discussing your answers to both questions with your teacher.

Discuss with your group, write, then describe to your teacher, how the accuracy and loss evaluated on the training and validation sets progress differently across epochs. Note the details. What does this tell us about the generalisation of the model? (**Question 2**, 5 points). So, make *at least 5 points* here.

Evaluate the model performance on the test set using the following command:

```
loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
```

Check that you see a loss of around 0.28 and an accuracy of around 0.92.

Discuss with your group, write, then describe to your teacher, whether you think this accuracy is sufficient for some uses of automatic hand-written digit classification you gave in Question 1. Why do you think this, considering how this digit classification would work in practice? (**Question 3**, 5 points).

Changing model parameters

Write your answers to questions 4-5 (below) together before discussing your answers to both questions with your teacher.

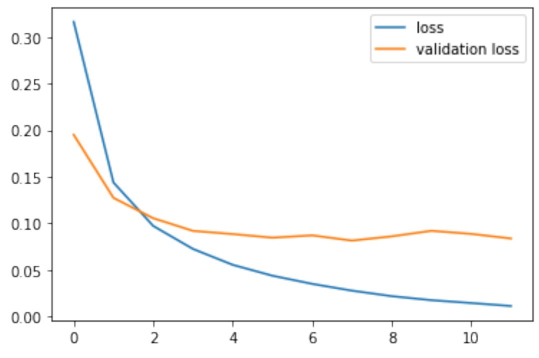
In the previous model, we did not specify an activation function for our hidden layer, so it used the default linear activation. Discuss with your group, write, then describe to your teacher, how linear activation of units limits the possible computations this model can perform. (**Question 4**, 3 points)

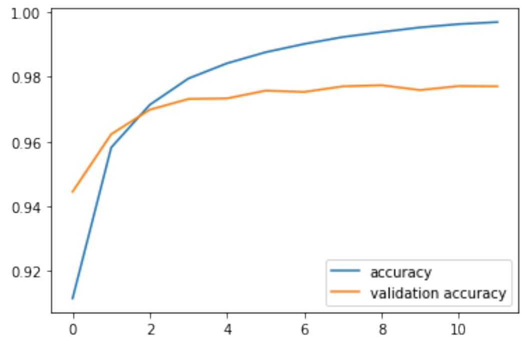
Now make a similar model with a rectified activation in the first hidden layer, by adding the extra argument:

```
activation = “relu”
```

to the model definition for this layer. Then compile, fit and evaluate the model.

Plot the training history as before. You should see something like the images below. Note the different values on the y-axis, compared to the previous plot.





Discuss with your group, write, then describe to your teacher, how this training history differs from the previous model, for the training and validation sets. Note the details.
Explain what this tells us about the generalisation of the model.
(**Question 5**, 6 points)

Deep convolutional networks
NEW TOPIC
Our first two models used fully-connected networks for number recognition. They ran quickly, largely because of their very simple structure. They learned the relationships between our pixels and the numbers they represent fairly well. However, they had a limited ability to generalise to new data that they were not trained on.

In deep convolutional networks, each convolutional filter samples from the limited space in the previous layer's feature map. To use the whole image to determine outputs, they need more layers to allow more spatial integration. They also need multiple feature maps at each layer to capture the multiple meaningful spatial relationships that are possible. All of this greatly increases computational load.

First, we need to prepare our data differently. Convolutional layers don't flatten x and y spatial dimensions, and need an extra dimension for colour channels (in the input image) or multiple feature maps (from previous convolution steps).

Reshape x_train to size 60000, 28, 28, 1. Reshape x_test to size 10000, 28, 28, 1. Rescale both results to values between zero and one as before.
y_train and y_test are categorical units, as before.

Now we will define a convolutional learning model with 2 convolutional layers that result from 32 convolutional filters into the first layer and 64 filters into the second. We will use 3x3 pixel filters to sample from the image to the first layer, and the same to sample from the first layer to the second. We will use rectified activation functions for both convolutional layers. We will use pooling to downsample the second convolutional layer to half its size in both spatial dimensions (so one quarter of the pixels). We will flatten the resulting feature map to one dimension, then use one fully-connected layer to link our network to the labels.

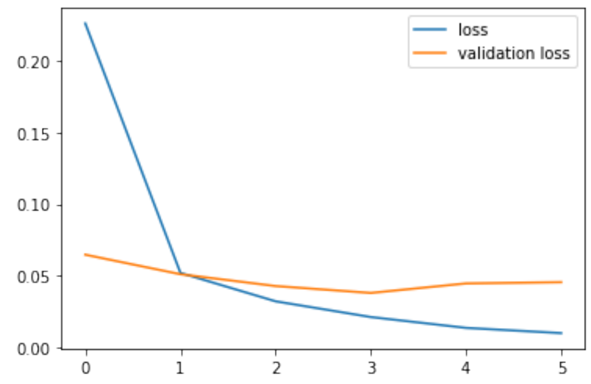
To make the (slightly deep) convolutional network model described above, enter the following code:
```
model = keras.Sequential()
model.add(keras.layers.Conv2D(filters=32, kernel_size=(3, 3),
activation="relu", input_shape=(28, 28, 1)))
model.add(keras.layers.Conv2D(filters=64, kernel_size=(3, 3),
activation="relu"))
model.add(keras.layers.MaxPool2D(pool_size=(2, 2)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(128, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
```

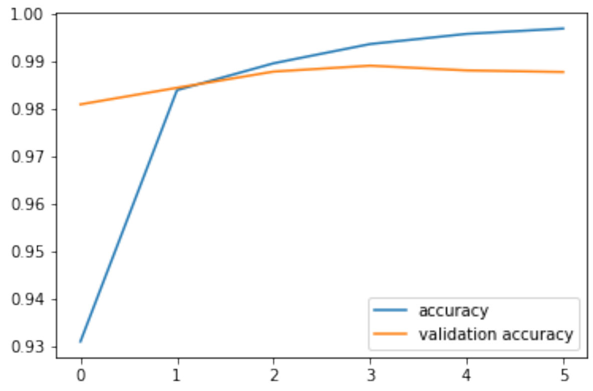
Now summarise the model to check it is what you want.
When compiling the model, we will use a slightly different backpropagation of error procedure, so change the 'optimizer' part of the compile command to:

```
model.compile(loss='categorical_crossentropy',
optimizer=keras.optimizers.Adadelta(learning_rate=float(1)),
metrics=[‘accuracy’])
```

Fit the model as before, but using only 6 epochs. Expect that model fitting will take far longer.

Plot the training history as before. You should see something like the images below. Again, note the different values on the y-axis, compared to the previous plots.





Again, consider how the training history differs from the previous model, for the training and validation sets, and what this tells us about the generalisation of the model. (You don't need to write this down, just think about it).

Do questions 6-7 (below) together before discussing your answers to these questions with your teacher.

Quantify the model's accuracy and loss as before.
Discuss with your group, write, then describe to your teacher, whether you think this is sufficient for the uses of automatic hand-written digit classification you listed in Question 1, and why. (**Question 6**, 5 points)

'Dropout' is a method used in deep network training to prevent overfitting of training data and focus on aspects of the learning model that will generalise to new data. (The challenge-based exercise asks you to research this in more detail).

Add dropout layers (`model.add(keras.layers.Dropout(rate=x))`) after the max pooling stage (x = 0.25) and after the fully-connected (dense) layer (x = 0.5). Compile and train the resulting model as before.

In comparison to the previous (convolutional) model:

a) How does the time taken for each training epoch differ? (Google Colab may not show this clearly, in which case run this on your own hardware)
b) How does the training history differ for both the training and validation sets?
c) How well does the resulting model generalise?
(**Question 7**, 7 points)

Here is a good point to start work on the challenge-based exercise, in pairs. But will not get help from your teachers on that, so class time is better spent working on Exercise Two, below.

Exercise Two: Low-level functions

NEW TOPIC

Deep learning relies on a few basic operations: convolution, nonlinear activation, pooling, and normalisation. A fully-connected layer linked to output nodes with a softmax function is also required. So far, we have used calls to Keras's high-level libraries to do these operations, as these are efficiently coded and easy to use. But it is also important that you understand the basic functions at a low level.

In each of the following questions, you will receive the highest points for smart and efficient code. Such code generally avoids working through each element in turn using 'for' loops, while also understanding where loops are necessary. Try instead to use matrix operations where possible, often combined with matrix reshaping operations.

Do not use high level functions like 'convolve' or 'flatten'. The aim is to implement these yourself, so you will not get credit for using existing implementations.

Questions 8-13 can be completed with the following low-level functions and operations only. If you are unfamiliar with any of these, look them up first:

- =, >, <, +, -, /, *, @ (@ is matrix multiplication, np.matmul is similar)
- reshape (this is very useful, particularly for efficient code)
- np.zeros (to pre-define arrays)
- np.sum
- np.maximum
- np.shape
- np.tile or np.repeat
- np.array
- np.mean
- np.std
- np.dot
- np.exp
- len
- range
- int
- for

Write a simple function that achieves the convolution operation efficiently for two-dimensional and three-dimensional inputs. This should allow you to input a set of convolutional filters ('kernels' in Keras's terminology) and an input layer (or image) as inputs. The input layer should have a third dimension, representing a stack of feature maps, and each filter should have a third dimension of corresponding size. The function should output a number of two-dimensional feature maps corresponding to the number of input filters, though these can be stacked into a third dimensional like the input layer.

In your answer, please give the code you have written (yourselves). Also include an example input image (for example from the mnist set) and the same image convolved with two 3x3 filters, one detecting horizontal edges and the other detecting vertical edges. (**Question 8**, 4 points). Make sure to keep the output for each filter separate.

Write a simple function that achieves rectified linear (relu) activation over a whole feature map, with a threshold at zero. In your answer, give the code you have written and one of the previous convolved images after this operation. (**Question 9**, 2 points)

Write a simple function that achieves max pooling. This should allow you to specify the spatial extent of the pooling, with the size of the output feature map changing accordingly. In your answer, give the code you have written and the image from the previous question after pooling with a 2x2 kernel. (**Question 10**, 3 points)

Write a simple function that achieves normalisation within each feature map, modifying the feature map so that its mean value is zero and its standard deviation is one. In your answer, give the code you have written. (**Question 11**, 2 points)

Write a function that produces a fully-connected layer. This should allow you to input the activations of every node in a stack of feature maps and a set of weights linking every input node to every output node. Then the size of the output activation array is already given in the sizes of these inputs. You should probably begin by flattening the stack of feature maps into a 1-dimensional matrix. In your answer, give the code you have written. (**Question 12**, 4 points)

Write a function that converts the activation of a 1-dimensional matrix (such as the output of a fully-connected layer) into a set of probabilities that each matrix element is the most likely classification. This should include the algorithmic expression of a softmax (normalised exponential) function. In your answer, give the code you have written. (**Question 13**, 2 points)

Exercise Three: Exam study

The following topics we have not studied in class because they are complex to explain (particularly as we are not computer scientists or mathematicians). We find students understand them better through self-study, and study to follow their own level of interest. You may like to do this in small groups. These topics **WILL** be included in the mid-term exam.

Backpropagation of error:
So far, we have avoided explaining backpropagation in detail. It is hard to follow such mathematical content in lectures. We have discussed what backpropagation does, and used Keras's implementation to train our networks. But we have not looked at how it works because the mathematics are complex and do not fit well with the goals of this course. However, this is a major principle in machine learning with neural networks. In the exam, we will ask you to explain in plain English (with minimal mathematical content) what backpropagation of error tries to achieve and/or how it does so.

To start studying this, we suggest watching videos 3 and 4 from the playlist at the following link. You may like to start by watching videos 1 and 2, as these set things up for videos 3 and 4:
https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

ChatGPT is also good at explaining backpropagation of error in plain English, which may help you study. But remember that ChatGPT's answers are often factually inaccurate so it is important to check the accuracy of its statements.

Text-to-image generation models:
An exciting recent advance in AI is the ability to generate realistic images that conform to text prompts, using text-to-image models. There are two central steps to this: the first step encodes the text into an embedding (so that words with similar meanings are represented similarly) while the second step generates an image that is consistent with the text's meaning. The overall process relies on several important types of machine learning network working together, so understanding this will give you a much deeper understanding of current AI networks.

Study the principles of the text-to-image generation process that is used in a powerful open-source text-to-image model: Stable Diffusion. DALL-E works by similar principle, but it is not open source, so we don't entirely know how it works.

You can find a straightforward introduction here:
https://jalammar.github.io/illustrated-stable-diffusion/

In the exam, we will ask you to explain in plain English the text-to-image generation process or specific central concepts in this process, such as:
-Transformer language encoders
-Diffusion models
-Latent space representations
-Image autoencoders
Again, exam questions will focus more on what the process tries to achieve and the principles by which it does so. Technical and mathematical aspects are less important.