TRANSFER LEARNING: TO CREATE A PRE-TRAINED MODEL

PEGHEADS.ORG
GLOBAL DIGITAL AI EDUCATIONAL ORGANIZATION

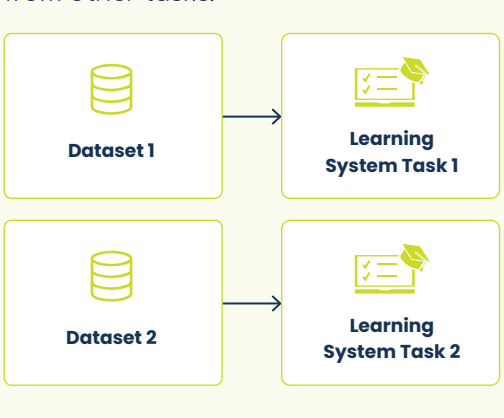
GOAL Train a pre-trained model with our dataset and create a pre-trained model of our own to do our task.

What is transfer learning? Transfer learning is a machine learning technique that enables data scientists to benefit from the knowledge gained from a previously used machine learning model for a similar task. An example would be using the knowledge gained while learning to classify cars to recognize the birds in the sky.

TRADITIONAL ML VS TRANSFER LEARNING

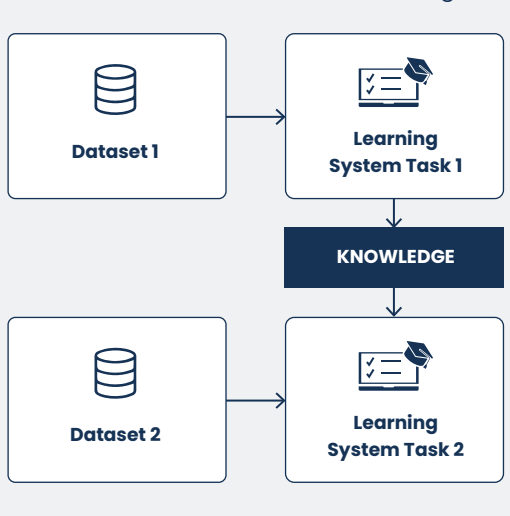
TRADITIONAL ML

- Isolated, single task learning.
- Knowledge is not retained or accumulated. Learning is performed w.o. consideration for knowledge learned from other tasks.

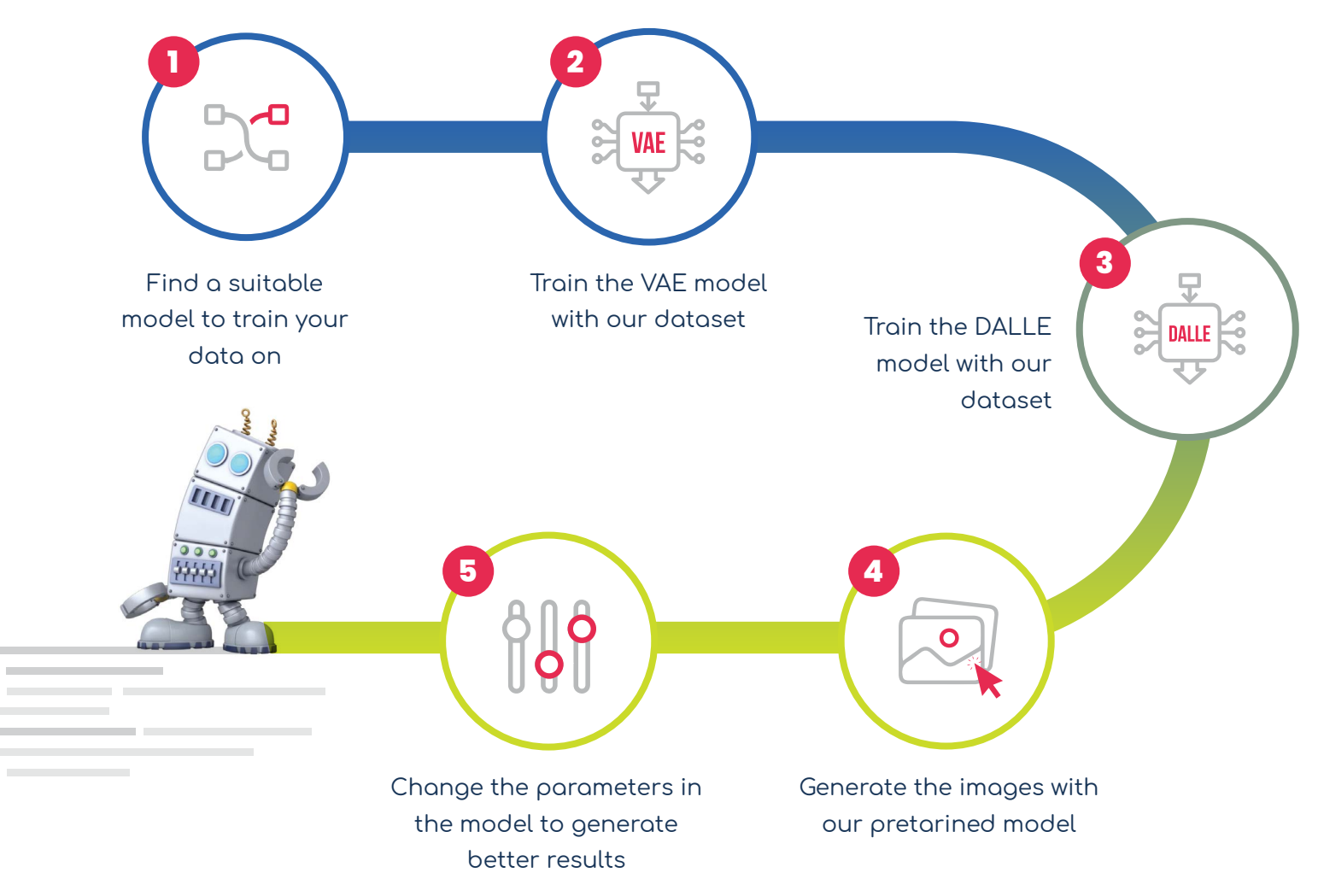


TRANSFER LEARNING

- Learning new tasks relies on previously learned tasks.
- Learning process can be faster, more accurate and/or need less training data.



When to use transfer learning? When there is not enough data and when there is not enough time to train the model



STEP 1

Find a suitable model to train your data on

We have selected the VAE and DALLE model.

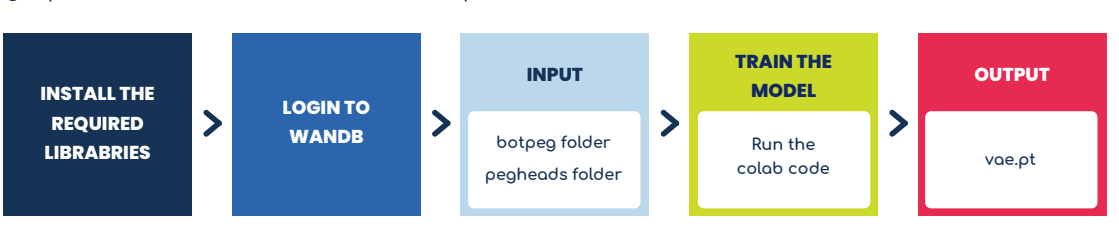
DALLE is a 12-billion parameter version of **GPT-3** trained to generate images from text descriptions, using a dataset of text–image pairs. But we are training it with our peghead images so that we get better results.

When we tried to generate images without training DALLE with the pegheads we did not get any output.

STEP 2

Train VAE model with our images and text

A variational autoencoder, also known as a VAE, is the artificial neural network architecture introduced by Diederik P. Kingma and Max Welling, belonging to the families of probabilistic graphical models and variational Bayesian methods.



INSTALL THE REQUIRED LIBRARIES

Install the following libraries:

`dalle-pytorch`	This library has the required dalle model and methods for training the model with our images
`wandb`	This will help you connect with the wandb database where the results can be stored.
`gdown`	This will help you to download the input files from your drive into the collab

LOGIN TO WANDB

Steps for registering are as follows:

Go to **wandb.ai/site** click sign up. now you can sign up with google or GitHub or email and password.

After signing up, you will have to fill out some information about yourself.

Once you finish you will be taken to the home page and can copy wandb API key. Don't share with anyone but you can reset it.

Feel free to take look at docs: **https://docs.wandb.ai/quickstart**

Run the following command to install and login to wandb in collab

```
import wandb
!wandb login
```

INPUT

We mainly use 2 folders.

The explanation about each file and the contents of it are as follows:

Folder1-name: **botpeg**

Folder or Filename	Contents	
`attributes`	Certinities.txt, class_attribute_labels_continuous.txt, image_attribute_labels.txt	
`images`	The images are organized in subdirectories based on Types of emotion or action	
`parts`	part_click_locs.txt, part_locs.txt, parts.txt	
`bounding_boxes.txt`	Each image contains a single bounding box label. with each line corresponding to one image: where `<image_id>` corresponds to the ID in images.txt, and `<x>`, `<y>`, `<width>`, and `<height>` are all measured in pixels	`<image_id> <x> <y> <width> <height>` consider the following example for the same: `1 195 195 580 400 here, image_id=1,x=195,y=195 width=580,height=400`
`classes.txt,`	The list of class names (botpeg's actions and emotions) is contained in the file with each line corresponding to one class:	`<class_id> <class_name>` consider the following example for the same: `1 Dance_01 here class_id=1 and class_name=Dance_01`
`image_class_labels.txt`	The ground truth class labels (botpeg labels) for each image are contained in the file, with each line corresponding to one image: where `<image_id>` and `<class_id>` correspond to the IDs in images.txt and classes.txt, respectively.	`<image_id> <class_id>` consider the following example for the same: `1 1 here image_id=1 and class_id=1`
`images.txt`	The list of image file names is contained in the file , with each line corresponding to one image	`<image_id> <image_name>` consider the following example for the same: `1 Dance_01.jpg here image_id=1 and image_name=Dance_01.jpg`
`train_test_split.txt`	The suggested train/test split is contained in the file with each line corresponding to one image: where `<image_id>` corresponds to the ID in images.txt, and a value of 1 or 0 for `<is_training_image>` denotes that the file is in the training or test set, respectively.	`<image_id> <is_training_image>` consider the following example for the same: `1 1 here image_id=1 and is_training_image=1`

`attributes`

Folder or Filename	Contents	
`Certinities.txt`	The list of all certainty names with each corresponding to one certainty	`<certainty_id> <certainty_name>` consider the following example for the same: `1 not visible here certainty_id=1 and certainty_name=not visible`
`class_attribute_labels_continuous.txt`	Each line corresponds to one class (in the same order as classes.txt) and each column contains one real-valued number corresponding to one attribute (in the same order as attributes.txt). The number is the percentage of the time (between 0 and 100) that a human thinks that the attribute is present for a given class	Our prgheads have 13 classes in total and 16 attributes hence. This will have 16 columns for each attribute and 13 rows for each class in case of our peghead folder
`image_attribute_labels.txt`	The set of attribute labels for each image is contained in the file where `<image_id>`, `<attribute_id>`, `<certainty_id>` correspond to the IDs in images.txt, attributes/ attributes.txt, and attributes/ certainties.txt respectively. `<is_present>` is 0 or 1 (1 denotes that the attribute is present). `<time>` denotes the time spent to label in seconds.	`<image_id> <attribute_id> <is_present> <certainty_id> <time>` consider the following example for the same: `1 1 1 4 45 here image_id=1,attribute_id=1,is_present=1,certainty_id=4,time=45`

`Parts`

Folder or Filename	Contents	
`part_click_locs.txt`	A set of multiple part locations for each image, with each line corresponding to the annotation of a particular part in a particular image. where `<image_id>`, `<part_id>`, `<x>`, `<y>` are in the same format as defined in parts/ part_locs.txt, and `<time>` is the time in seconds spent to label	`<image_id> <part_id> <x> <y> <visible> <time>` consider the following example for the same: `1 1 0 0 0 1 here image_id=1,part_id=1,x=0,y=0,visible=0,time=1`
`part_locs.txt`	The set of all ground truth part locations is contained in the file parts/ part_locs.txt, with each line corresponding to the annotation of a particular part in a particular image: where `<image_id>` and `<part_id>` correspond to the IDs in images.txt and parts/parts.txt, respectively. `<x>` and `<y>` denote the pixel location of the center of the part. `<visible>` is 0 if the part is not visible in the image and 1 otherwise.	`<image_id> <part_id> <x> <y> <visible>` consider the following example for the same: `1 1 0 0 0 here image_id=1,part_id=1,x=0,y=0,visible=0`
`parts.txt`	The list of all part names is contained in the file parts/parts.txt, with each line corresponding to one part:	`<part_id> <part_name>` consider the following example for the same: `1 left antenna here part_id=1,part_name=left antenna`

`attributes.txt`

The list of all attribute names is contained in the file attributes/attributes.txt, with each line corresponding to one attribute:

`<attribute_id> <attribute_name>`

consider the following example for the same:

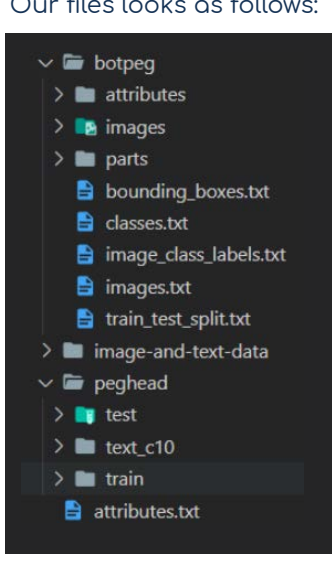
`1 botpeg's_hand_curved::up`

`here attribute_id=1 and attribute_name=botpeg's_hand_curved::up`

Folder2: **peghead**

Folder or Filename	Contents
`train`	has all the images and text files to train the model
`test`	has all the images and text files to test the model
`text_c10`	it has some sample texts

Our files looks as follows:



Convert these into a zip folder and save it in your drive.

Note: This step is done to ensure that the data you are training the model with is organized properly .also if you have a huge amount of data you can zip it else you can load them directly to the colab and skip the zipping part.

https://colab.research.google.com/drive/1RkTVfcRDbvcPb0uUIVdh2EQWoR-PP2KZ?usp=sharing

STEP 3

Train the DALLE model with Peghead images and test it to see if it generates any output.

INPUT

In the previous step we got vae.pt as our output, it's the VAE model trained with our peghead images, we will use this as an input to our DALLE model along with a folder named image-and-text-data, which contains the images and text files. You can use the same files which you used for training and testing the VAE model.

For example:

BotPeg Singing

Dalle uses the simple tokenziser to tokenize the text inputs and then learns how to generate images from the given text.

RUN THE CODE

You can use the following code to train DALLE with your images.

https://github.com/PegHeads-Inc/PegHeads-Tutorial-4//blob/main/train_dalle.py

Once you run the code you will get a trained dalle model which will have learnt what is a botpeg.

Please note once you run the code in visual studio make sure that you clear you system cache memory because it fills it up with the wandb results and makes the system slow.

OUTPUT

A file called dalle.pt will be generated that is you pretrained DALLE model

STEP 4

Generate the images with our pretarined model

Next you can use the code below to generate the images (generate.py)

https://github.com/PegHeads-Inc/PegHeads-Tutorial-4/blob/main/generate.py

Note: We trained our model with 20 epochs first and the results we blur.

STEP 5

Change the parameters in the model to generate better results

So we tried training it with 200 epochs and a batch size of 1. And we got better results as shown below.

Check out our git repository for the entire code.

STAY TUNED FOR OUR NEXT TUTORIAL

If you would like to contribute to this tutorial with any information or tests and upload to our GitHub please do. This is an open platform to help anyone learn AI and machine learning skills.

https://github.com/PegHeads-Inc/PegHeads-Tutorial-4/issues

PEGHEADS.ORG
GLOBAL DIGITAL AI EDUCATIONAL ORGANIZATION