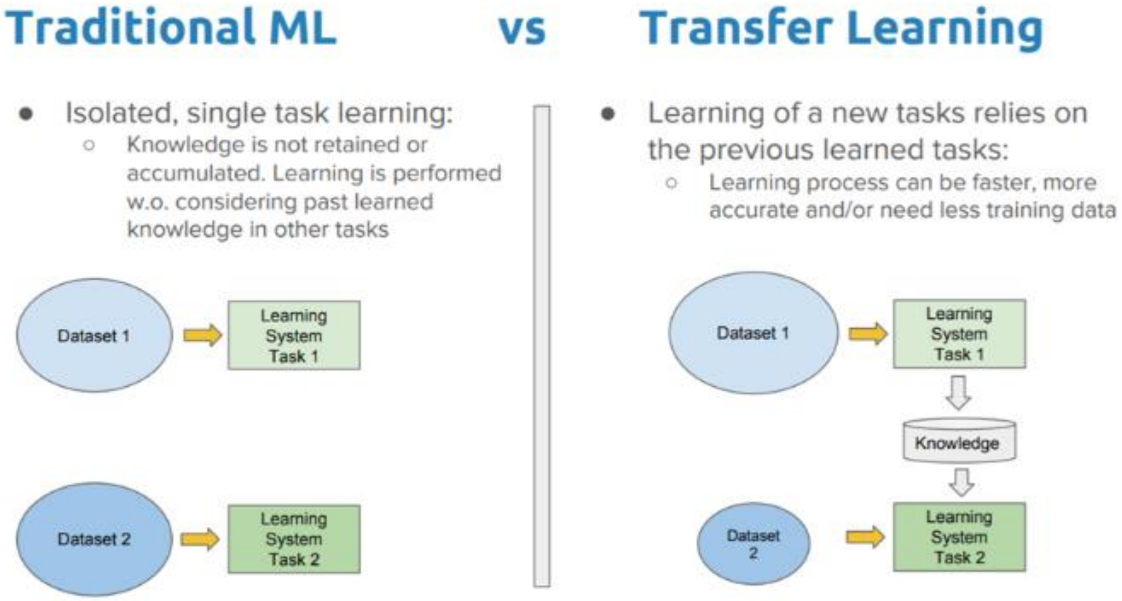
Machine learning Tutorial using Transfer Learning

Traditional Machine learning is isolated and occurs purely based on specific tasks, datasets and training separate isolated models on them. No knowledge is retained which can be transferred from one model to another. In transfer learning, you can leverage knowledge (features, weights etc) from previously trained models for training newer models and even tackle problems like having less data for the newer task. Advantages of using transfer learning is that it saves lot of time while training as the entire dataset need not to be trained again.



Traditional ML vs Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks

- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data

Traditional Learning vs Transfer Learning

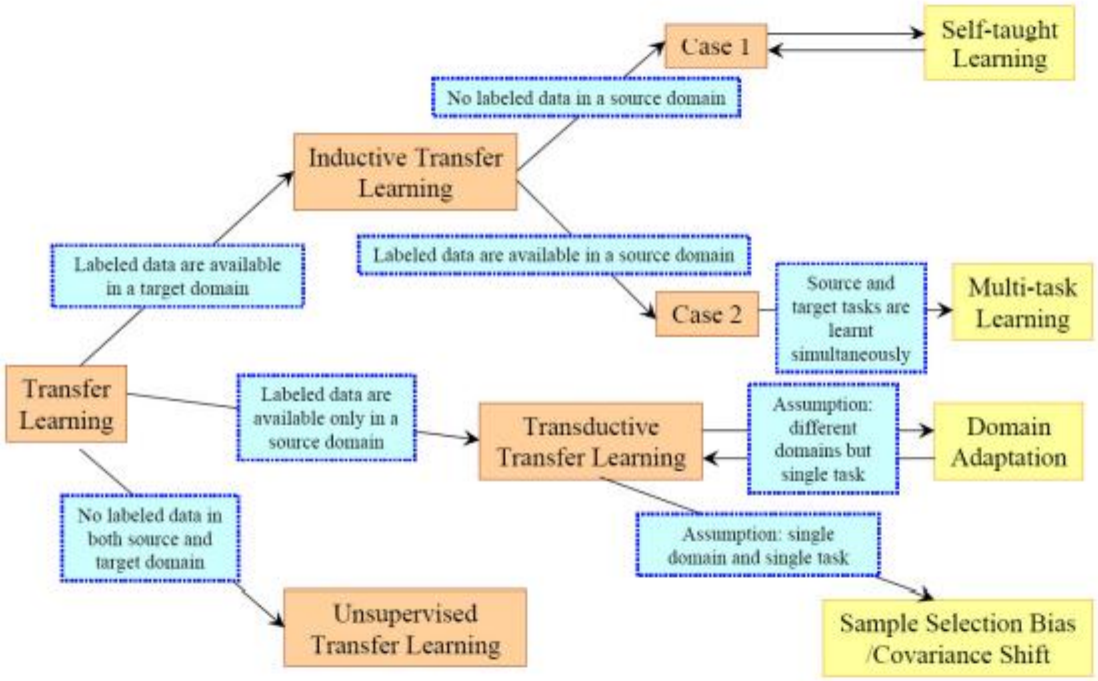
During the process of transfer learning these are the most important questions that comes into mind:

- **What to transfer:** This is the first and the most important step in the whole process. We try to seek answers about which part of the knowledge can be transferred from the source to the target in order to improve the performance of the target task. When trying to answer this question, we try to identify which portion of knowledge is source-specific and what is common between the source and the target.

- **When to transfer:** There can be scenarios where transferring knowledge for the sake of it may make matters worse than improving anything (also known as negative transfer). We should aim at utilizing transfer learning to improve target task performance/results and not degrade them. We need to be careful about when to transfer and when not to.

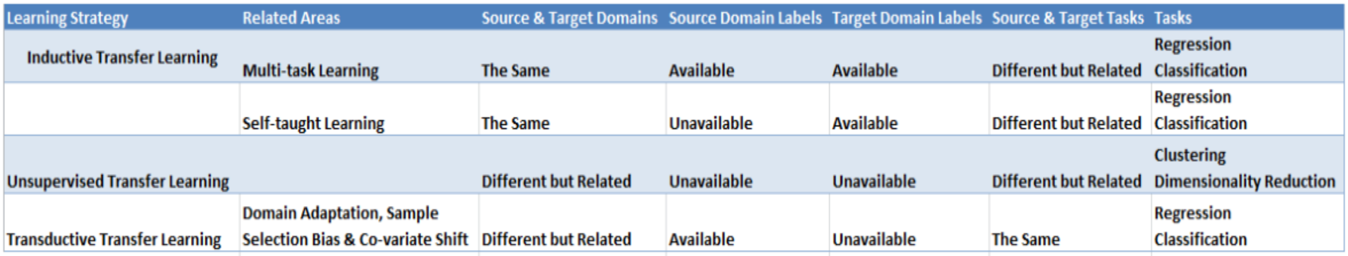
- **How to transfer:** Once the *what* and *when* have been answered, we can proceed towards identifying ways of actually transferring the knowledge across domains/tasks. This involves changes to existing algorithms and different techniques, which we will cover in later sections of this article. Also, specific case studies are lined up in the end for a better understanding of how to transfer.

There are various different strategies that can be applied to transfer learning process based on the domain, task at hand and availability of data. This is figure from the survey paper on machine learning which helped a lot during the project development.


Case 1
Self-taught Learning
No labeled data in a source domain
Inductive Transfer Learning
Labeled data are available in a target domain
Labeled data are available in a source domain
Case 2
Source and target tasks are learnt simultaneously
Multi-task Learning
Transfer Learning
Labeled data are available only in a source domain
Transductive Transfer Learning
Assumption: different domains but single task
Domain Adaptation
No labeled data in both source and target domain
Assumption: single domain and single task
Unsupervised Transfer Learning
Sample Selection Bias /Covariance Shift

Transfer Learning Strategies

Summarizing the above flow diagram in the table below:



Learning Strategy	Related Areas	Source & Target Domains	Source Domain Labels	Target Domain Labels	Source & Target Tasks	Tasks
Inductive Transfer Learning	Multi-task Learning	The Same	Available	Available	Different but Related	Regression Classification
	Self-taught Learning	The Same	Unavailable	Available	Different but Related	Regression Classification
Unsupervised Transfer Learning		Different but Related	Unavailable	Unavailable	Different but Related	Clustering Dimensionality Reduction
Transductive Transfer Learning	Domain Adaptation, Sample Selection Bias & Co-variate Shift	Different but Related	Available	Unavailable	The Same	Regression Classification

Types of Transfer Learning Strategies and their Settings

Detailed steps involved on how to run a machine learning project from scratch with all the linux commads.

Step1: Create a base directory to store the files required.

```
mkdir asl_project
```

Step 2: Go to the created directory.

```
cd asl_project
```

Step 3: create the following sub directories.

```
mkdir logs

mkdir dataset
```

Step 4: if you would like to use previously trained graph provided by us, skip to step 9 else continue.

Step 5: copy the images that you would like to use to train the model inside the dataset folder created in step 3. You have to copy minimum 100 different images showing each letter in specific folders.

Step 6: copy the python script file "train.py" in the folder asl_project created in step 1.

Step 7: use the following command to train the model using dataset created in step 5.

```
python3 train.py \
  --bottleneck_dir=logs/bottlenecks \
  --how_many_training_steps=2000 \
  --model_dir=inception \
  --summaries_dir=logs/training_summaries/basic \
  --output_graph=logs/output_graph.pb \
  --output_labels=logs/output_labels.txt \
  --image_dir=./dataset
```

Step 8: the above step will generate the trained graph and trained labels file inside the logs folder created in step 3
Step 9: skip step 8 and continue from step 10.

Step 9: copy the files "output_labels.txt" and "output_graph.pb" into logs folder created in step 3.

Step 10: copy the classification python script "classify.py" into asl_project" folder created in step 1

Step 11: test the classification using the following command:

```
python3 classify.py path/to/image.jpg
```

Step 12: if the classification works perfectly in step 11, proceed to step 13.

Step 13: copy the “webcam.py” script to asl_project folder created in step 1.

Step 14: use the following command to use the webcam and detect the gestures:

```
Python3 webcam.py
```

Please note that Your hand must be inside the rectangle.