

# A Hands-on Introduction to Transfer Learning

Tamoghna Ghosh, AI Solutions Architect , Intel (Client Computing Group)

# Agenda

Module 1: Theory of Transfer Learning

Module 2: Applications of TL: Images

Module 3: Applications of TL: Text

Module 4: Application of TL: Audio

Module 5: Advanced Application of TL

# Module 1: Theory of Transfer Learning

What, when and How to transfer

# Module 1: Detailed Agenda

- What is transfer learning?
- When to use transfer learning?
- Formal definition of transfer learning.
  - Mathematical Formulation
  - Formally, when to use transfer learning
  - Types of Transfer learning
- How to Do transfer learning in various scenarios
- Challenges of Transfer learning
- Presentation Time : 20 minutes

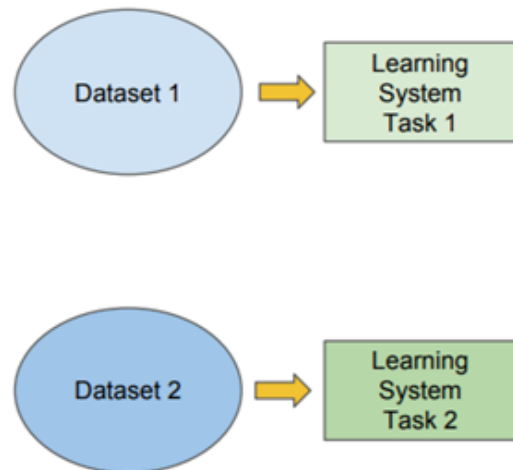
# Transfer Learning

- To learn a new task, we humans need not always start afresh but rather apply previous learned knowledge. Any new concept is not learned in isolation, but considering connections to what is already known, which makes the skill of building analogies one of the cores of human intelligence
- Can Machines learn in the same way?
- "Transfer Learning" (TL) allows a machine learning model to port the knowledge acquired during training of one task to a new task.

# How TL is different from traditional ML

## Traditional ML

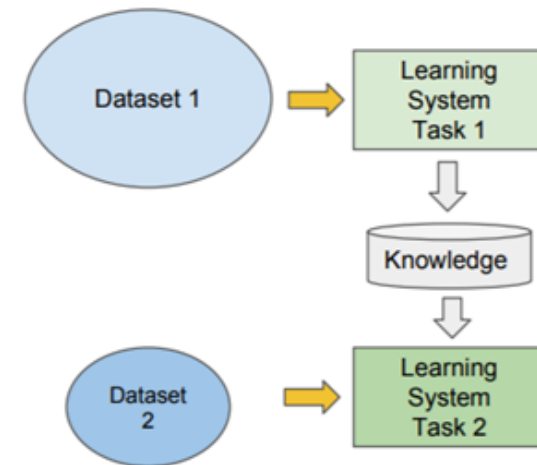
- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



vs

## Transfer Learning

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



# When to use transfer learning

---

TL is a mostly used to obtain well-performing machine learning model in settings where high-quality labelled data is scarce. Sometimes, training data is expensive or difficult to collect.

---

Even we have access to large dataset, but we have scarcity of resources like time or compute

---

Privacy Preserving ML: Train ML models without exposing private data in its original form.

---

Multimodal Deep Learning: Data has multiple modalities (audio, video, text, structured) and we need to use all forms of data to build a model.

# Formal Definition of Transfer Learning

- Notations
  - $\mathcal{D}$  – Domain
  - $\mathcal{X}$  –  $n$  dimensional Feature Space
  - $\mathcal{Y}$  – Target Space ( discrete or continuous)
  - $\mathcal{T}$  – Task
  - $f(.) \sim P(Y|X)$  - Predictive Function , where  $Y \in \mathcal{Y}$  and  $X \in \mathcal{X}$  –
- Domain  $\mathcal{D} = (\mathcal{X}, P(\mathcal{X}))$  , where  $P(\mathcal{X})$  represents probability distribution over  $\mathcal{X}$
- Task  $\mathcal{T} = (\mathcal{Y}, f(.))$



# Example to Demonstrate the Notations

- Consider the food review sentiment analysis problem
  - $\mathcal{D}$  – *Food Reviews*
  - $\mathcal{X}$  – *Tf-idf representation of the review text or some dense representation of the review text as fixed size vectors*
  - $\mathcal{Y}$  – *Discrete – Positive or Negative*
  - $\mathcal{T}$  – *Classification*
  - $f(\cdot)$  – *Predictive Binary classification model which takes a text representation from  $\mathcal{X}$  and outputs a probability for the positive review.*

# Transfer Learning

- Given a source domain  $\mathcal{D}_S$  with a corresponding source task  $\mathcal{T}_S$
- A target domain  $\mathcal{D}_T$  with a corresponding task  $\mathcal{T}_T$
- Transfer learning is the process of improving the target predictive function  $f_T$  by using the related information from  $\mathcal{D}_S$  and  $\mathcal{T}_S$ , where  $\mathcal{D}_S \neq \mathcal{D}_T$  or  $\mathcal{T}_S \neq \mathcal{T}_T$ .

# How Domains and Tasks Vary

- $\mathcal{D}_S \neq \mathcal{D}_T$ , where  $\mathcal{D}_S = (\mathcal{X}_S, P(\mathcal{X}_S))$  and  $\mathcal{D}_T = (\mathcal{X}_T, P(\mathcal{X}_T))$ 
  - Either  $\mathcal{X}_S \neq \mathcal{X}_T$ , Feature spaces are different
  - Or  $P(\mathcal{X}_S) \neq P(\mathcal{X}_T)$ , Feature space same but the distribution of features is different
- $\mathcal{T}_S \neq \mathcal{T}_T$ , where  $\mathcal{T}_S = (\mathcal{Y}_S, f_S(.))$  and  $\mathcal{T}_T = (\mathcal{Y}_T, f_T(.))$ 
  - Either  $\mathcal{Y}_S \neq \mathcal{Y}_T$ , label spaces are different
  - or  $f_S(.) \neq f_T(.)$  i.e., conditional probability distributions between the source and target domains are different
  - or  $P(\mathcal{Y}_S) \neq P(\mathcal{Y}_T)$  i.e., different target data distribution like imbalance in classes

# Scenarios When to use TL (Formal terms Source & Target Domains)

- **Unsupervised Transfer:** No labelled data in either source or target domain, but domains are similar with *unsupervised task* in the target domain. Eg:
  1. Cluster EMNIST letters only (source w/o labels) given MNIST (target w/o labels)
  2. Creating word Embeddings for target domain given very large source domain
- **Inductive Transfer:** Same source and target domains, but tasks are different. Use the knowledge from the source model to improve the target task model. Eg:
  1. Image Caption generator : source Flickr8k -> target: Flickr Outdoor Scenes with text describing scene
- **Transductive Transfer:** Similarities between the source and target tasks, but the corresponding domains are different. Source domain has a lot of labeled data, while the target domain has none. Eg:
  1. MNIST (source) -> USPS (Target w/o labels)
  2. Amazon Reviews(source)-> IMDB(target w/o labels)
  3. Audio Event Detection (target) using Imagenet Source

# Measuring Domain Closeness

- Domain/Task relatedness can be determined by simple reasoning, such as images from different viewing angles or different lighting conditions. Examples:
  1. In medical field, images captured from different devices
  2. In case of handwriting one domain of data may be handwriting on smart board whereas the other is handwriting on smart phone or tablet
  3. Food Reviews vs Hotel Reviews vs Movie Reviews.
- Formally:
  - Two tasks to be similar if they use the same features to make a decision.
  - Data for both domains are similar if they can be generated from a fixed probability distribution using a set of transformations.

# Mathematically Measuring Closeness of Domains

- Maximum mean discrepancy (MMD) is an index to measure the discrepancy of two distributions in terms of distances between *mean embeddings* of features.
- Given  $P(\mathcal{X}_S) \neq P(\mathcal{X}_T)$  and a **nonlinear mapping function**  $\phi: \mathcal{X} \rightarrow \mathcal{H}$  ( $\mathcal{H}$  is reproducing Kernel Hilbert space (RKHS)),

$$\text{MMD} = \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \phi(x_i^S) - \frac{1}{n_t} \sum_{i=1}^{n_t} \phi(x_i^T) \right\|_{\mathcal{H}}^2$$

Here,  $\mathcal{H}$  is a space of real valued functions defined on  $\mathcal{X}$ .

A function  $K: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  ( $K(x, y) = \phi(x)^T \phi(y)$ ) is called a kernel.  $\phi$  is called the feature map.  $K$  is called reproducing kernel if it satisfies the following properties:

- $\forall x \in \mathcal{X}, k(\cdot, x) \in \mathcal{H},$
- $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x)$  (the reproducing property).

# High Level Types of TL

- Homogeneous Transfer:
  - $\mathcal{X}_S = \mathcal{X}_T$  But  $P(\mathcal{X}_S) \neq P(\mathcal{X}_T)$
  - $\mathcal{Y}_S = \mathcal{Y}_T$  But  $P(\mathcal{Y}_S) \neq P(\mathcal{Y}_T)$
- Heterogeneous Transfer
  - $\mathcal{X}_S \neq \mathcal{X}_T$  And / OR
  - $\mathcal{Y}_S \neq \mathcal{Y}_T$ ,  $\mathcal{Y}_S$  and or  $\mathcal{Y}_S$  can be  $\emptyset$  (unsupervised)

# How to Transfer

## Transfer Learning Problems

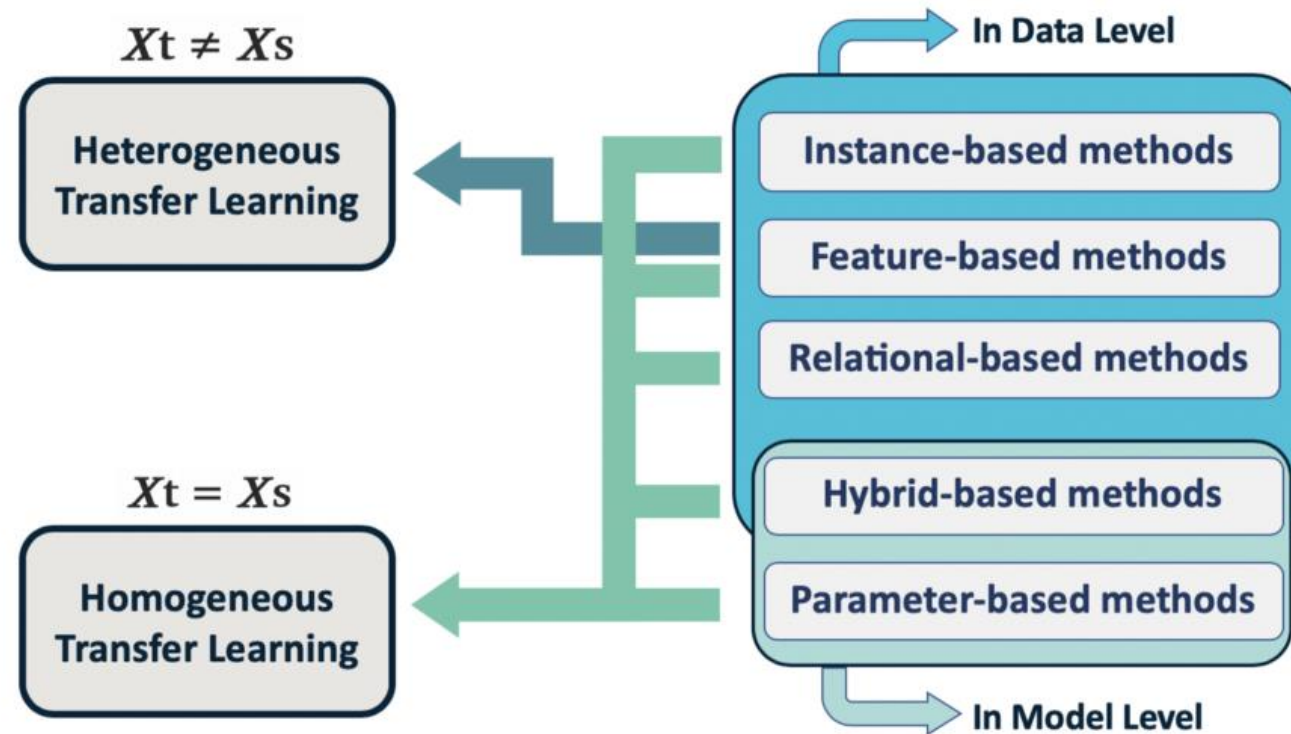
$X_t \neq X_s$

Heterogeneous  
Transfer Learning

$X_t = X_s$

Homogeneous  
Transfer Learning

## Transfer Learning Solutions



Source: <https://medium.com/georgian-impact-blog/transfer-learning-part-1-ed0c174ad6e7>



# Instance Based Transfer

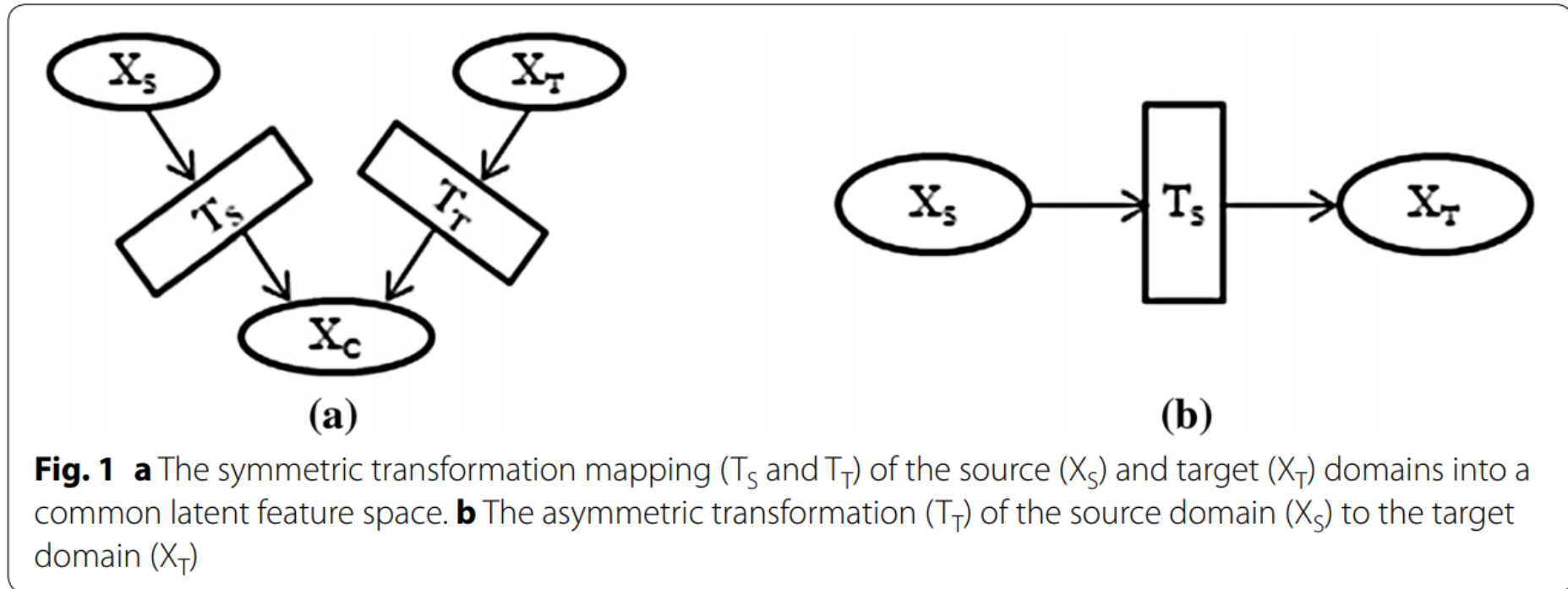
- To make a source domain trained model perform better, certain instances from the source domain can be reused along with target data to improve results.
  - AdaBoosting type approaches can be used to improve target model. Transfer AdaBoost learning framework TrAdaBoost, extends AdaBoost for transfer learning. It allows users to utilize a small amount of newly labeled data to leverage the old data to construct a high-quality classification model for the new data.
- This is extended to multi source transfer.

## References:

- Dai W, Yang Q, Xue GR, Yu Y (2007) Boosting for transfer learning. In: Proceedings of the 24th international conference on machine learning. p. 193–200.
- Yao Y, Doretto G. Boosting for transfer learning with multiple sources. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition. 2010. p. 1855–62.

# Feature Based Approach

- Applicable to both homogeneous and heterogeneous problems. There are 2 types:
- **Symmetric:** discovers underlying meaningful structures between the domains to find a common latent feature space Figure (a)
- **Asymmetric:** Transforms the features of the source to more closely match the target domain Figure (b)



# Parameter Based

A parametric model is learned in the source domain and transferred knowledge is encoded into parameters

- Multi-Source Parameter Transfer: If we have limited labelled target data and multiple labeled source domains, we can train separate classifiers for the source domains and optimally combine the re-weighted learners like ensemble learners to form an improved target learner.
- Single-Source Transfer: NN are parametric models, we can transfer the parameters learned in feature domain to train a classifier in target domain.

# Relation Based Transfer

- Relational-knowledge transfer attempts to handle non-i.i.d data.
  - where each data point has a relationship with other data points; for instance, social network data utilizes relational-knowledge-transfer techniques.
- Reference: GLoMo: Unsupervisedly Learned Relational Graphs as Transferable Representations
  - Instead of transferring features, we transfer the graphs output by a network. The graphs are multiplied by task-specific features (e.g. embeddings or hidden states) to produce structure-aware features

# Transfer Learning Strategies in Deep Learning(DL)

- Domain Adaptation: (Parameter or Asymmetric Feature based)
- Domain Confusion: (Symmetric Feature based)
- Extreme Variants of TL: Multi-Task Learning, Zero-shot and Few Shot (Symmetric feature Based)

# Challenges

- Learning without Forgetting
- Negative Transfer
- Security Vulnerabilities of Transfer Learning

# Module 2: Application of TL For Image Source and Target Domains

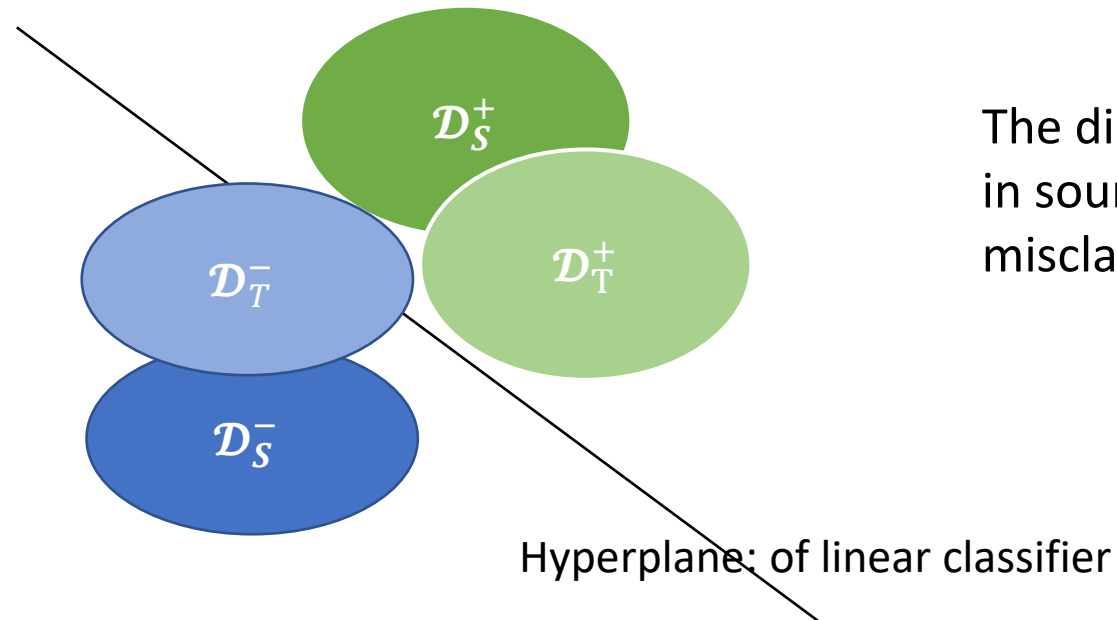
# Module 2: Detailed Agenda

- Asymmetric Feature Based Approach Domain Adaptation:
  - Adversarial Domain Adaptation: MNIST (source) -> USPS (Target w/o labels)
- Symmetric Feature Based Domain Confusion:
  - Joint Distribution Adaptation: MNIST (source) -> USPS (Target w/o labels)



# Case of No Labelled data in target domain Only: Feature based Transfer

- No Labelled data in target domain, enough labelled data in source domain. However, source and target domains have the same label spaces



The direct use of trained discriminative hyperplane in source domain will lead to the extensive misclassification in target domain.

# Joint Distribution Adaptation (JDA)

- First Introduced : Transfer Feature Learning with JDA(1CCV 2013) Cited by 1386
- Later Work: Deep Transfer Network with JDA by T Han · 2018 · Cited by 277
- USPS dataset consists of 7,291 training images and 2,007 test images of size  $16 \times 16$ . MNIST dataset has a training set of 60,000 examples and a test 28.

MNIST



USPS

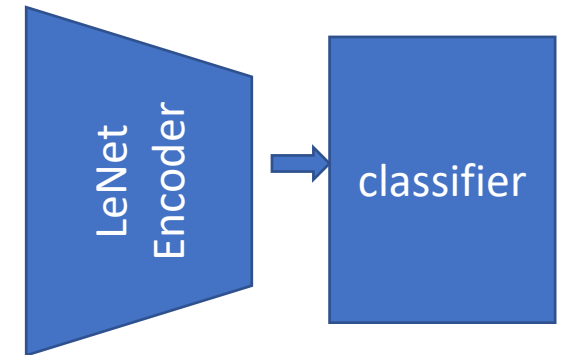


# JDA Loss

- JDA :  $\min D(P_S(\phi(X_S)), P_T(\phi(X_T)))$ ,  $D$  is the function to evaluate the domain discrepancy.
- JDA = MDA + CDA
- MDA = Marginal Distribution Adaptation : contributes to improving transfer performance – can be approximated by MMD Metric
- CDA = Conditional Distribution Adaptation: aims to match the discrimination structures between labeled source data and unlabeled target data.
  - Can be approximated by computing class wise MMD, where class labels in target are obtained using pseudo-labels predicted by source trained model

# Approach

- Domains: Source = MNIST , Target = USPS (w/o labels)
- Train Model on Source and Adapt to the target later
- Model:
  - Train Full model with cross-entropy loss on source
  - Joint Training on target and source with same cross-entropy loss on Source data batch and JDA loss on target. JDA loss requires class labels, we will use labels predicted by model and compute JDA loss for target data.
  - Encoder adapts symmetrically to both domains and tries to perform better in both domains.



# Notebook JDA

- [https://github.com/tamoghnaG/ODSC22-west-A-Hands-on-Introduction-to-Transfer-Learning/blob/main/Notebook 01 JDA.ipynb](https://github.com/tamoghnaG/ODSC22-west-A-Hands-on-Introduction-to-Transfer-Learning/blob/main/Notebook%2001%20JDA.ipynb)

# Adversarial Discriminative Domain Adaptation (ADDA)

- Unsupervised Domain Adaptation: there are no labelled observations in the target domain, but lots of training data in source domain.

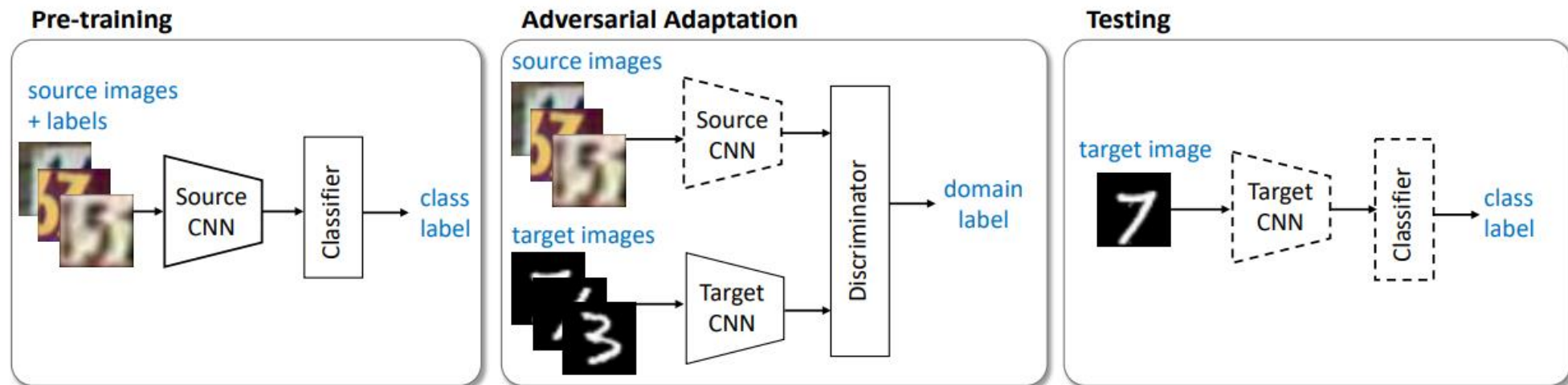


Figure 3: An overview of our proposed Adversarial Discriminative Domain Adaptation (ADDA) approach. We first pre-train a source encoder CNN using labeled source image examples. Next, we perform adversarial adaptation by learning a target encoder CNN such that a discriminator that sees encoded source and target examples cannot reliably predict their domain label. During testing, target images are mapped with the target encoder to the shared feature space and classified by the source classifier. Dashed lines indicate fixed network parameters.

# AADA: Approach

- Pretraining: Train CNN based Classifier for source domain
- TODO : GAN loss

# Notebook : ADDA

- [https://github.com/tamoghnaG/ODSC22-west-A-Hands-on-Introduction-to-Transfer-Learning/blob/main/Notebook 02 AADA.ipynb](https://github.com/tamoghnaG/ODSC22-west-A-Hands-on-Introduction-to-Transfer-Learning/blob/main/Notebook%20AADA.ipynb)



# Module 3: Transfer Learning in Text

# Module 3: Detailed Agenda

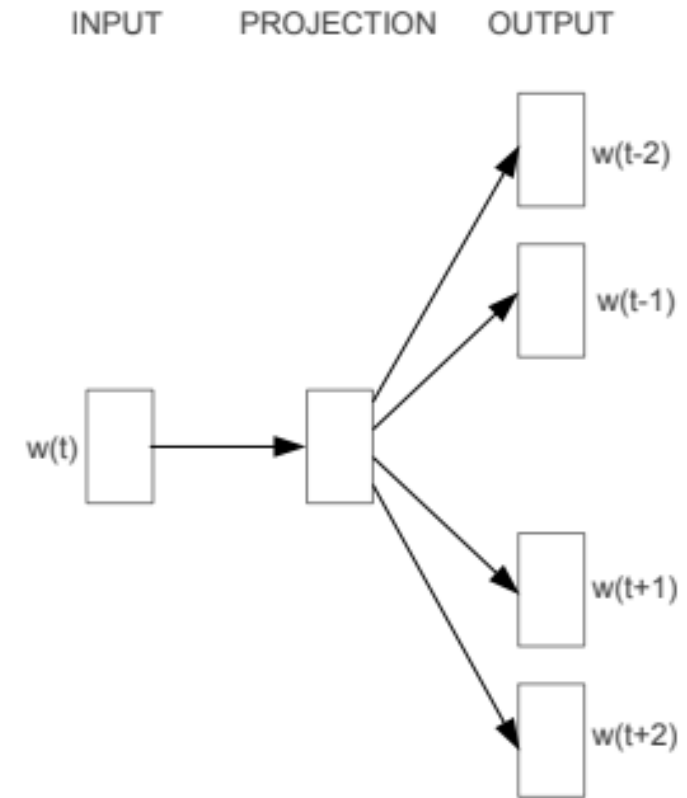
- Unsupervised Transfer: Unsupervised Text Representation or word embedding. Examples:
  - SkipGram with hands-on code
- Feature based asymmetric Transfer: Sentiment Analysis: Target domain has less labelled data, source has labelled data.
  - SA learned on Amazonreviews (product reviews) works on small movie review dataset with fine-tuning final layers of source classifier (Hands-on)
- Feature based Symmetric Transfer: Sentiment Analysis: Target domain has No labelled data source has labelled data.
  - SDAE : Domain Adaptation for Large-Scale Sentiment Classification (by X Glorot ICML 2011 · Cited by 2024)
  - BERT Pre-Trained with MLM on both datasets to get sentence embedding. Train classifier on product review and apply directly on movie review w/o further finetuning

# Text Pre-processing Steps

- Each Document Is Processed As Follows:
  - Tokenize document to Words
  - Lowercasing Words
- Build Vocab: Keep Words Appearing  $>$  MIN\_WORD\_COUNT

# Skip-Gram Model: Unsupervised Word Embedding

Window Size	Text	Skip-grams
2	[ The wide road shimmered ] in the hot sun.	wide, the wide, road wide, shimmered
	The [ wide road shimmered ] in the hot sun.	shimmered, wide shimmered, road shimmered, in shimmered, the
	The wide road shimmered in [ the hot sun ].	sun, the sun, hot
3	[ The wide road shimmered in ] the hot sun.	wide, the wide, road wide, shimmered wide, in
	[ The wide road shimmered in the hot ] sun.	shimmered, the shimmered, wide shimmered, road shimmered, in shimmered, the shimmered, hot
	The wide road shimmered [ in the hot sun ].	sun, in sun, the sun, hot



Skip-gram

# Learning SkipGram Embeddings

- Link To Notebook
- Learned Similar Words

<b>broken</b>	shattered	cracked	dented	chipped	damaged
<b>damaged</b>	defective	broken	dented	faulty	scratched
<b>awesome</b>	awsome	amazing	fantastic	excelent	great
<b>useful</b>	usefull	helpful	valuable	beneficial	specific
<b>good</b>	great	decent	bad	nice	ok
<b>easy</b>	difficult	simple	quick	convenient	hard
<b>violent</b>	brutal	misogynistic	sadistic	gory	vicious
<b>romantic</b>	romance	fairytale	comedy	screwball	erotic
<b>nasty</b>	freaky	boob	decapitation	sickening	gross
<b>unfortunate</b>	obvious	chiefly	dubious	redemptive	loathed
<b>predictable</b>	implausible	formulaic	clichéd	contrived	talky
<b>hilarious</b>	funny	hysterical	amusing	hilarious	charming

# Asymmetric Transfer

- 1D CNN based SA learned on Amazonreviews (product reviews) works on small movie review dataset with fine-tuning final layers of source classifier and document embedding layer (Hands-on)
- Notebook Link: [https://github.com/tamoghnaG/ODSC22-west-A-Hands-on-Introduction-to-Transfer-Learning/blob/main/Notebook SA 03.ipynb](https://github.com/tamoghnaG/ODSC22-west-A-Hands-on-Introduction-to-Transfer-Learning/blob/main/Notebook%20SA%2003.ipynb)

# Symmetric Transfer

---

## Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach

---

Xavier Glorot<sup>(1)</sup>  
Antoine Bordes<sup>(2,1)</sup>  
Yoshua Bengio<sup>(1)</sup>

GLOROTXA@IRO.MONTREAL.CA  
BORDESAN@HDS.UTC.FR  
BENGIOY@IRO.MONTREAL.CA

<sup>(1)</sup> Dept. IRO, Université de Montréal, Montréal (QC), H3C 3J7, Canada

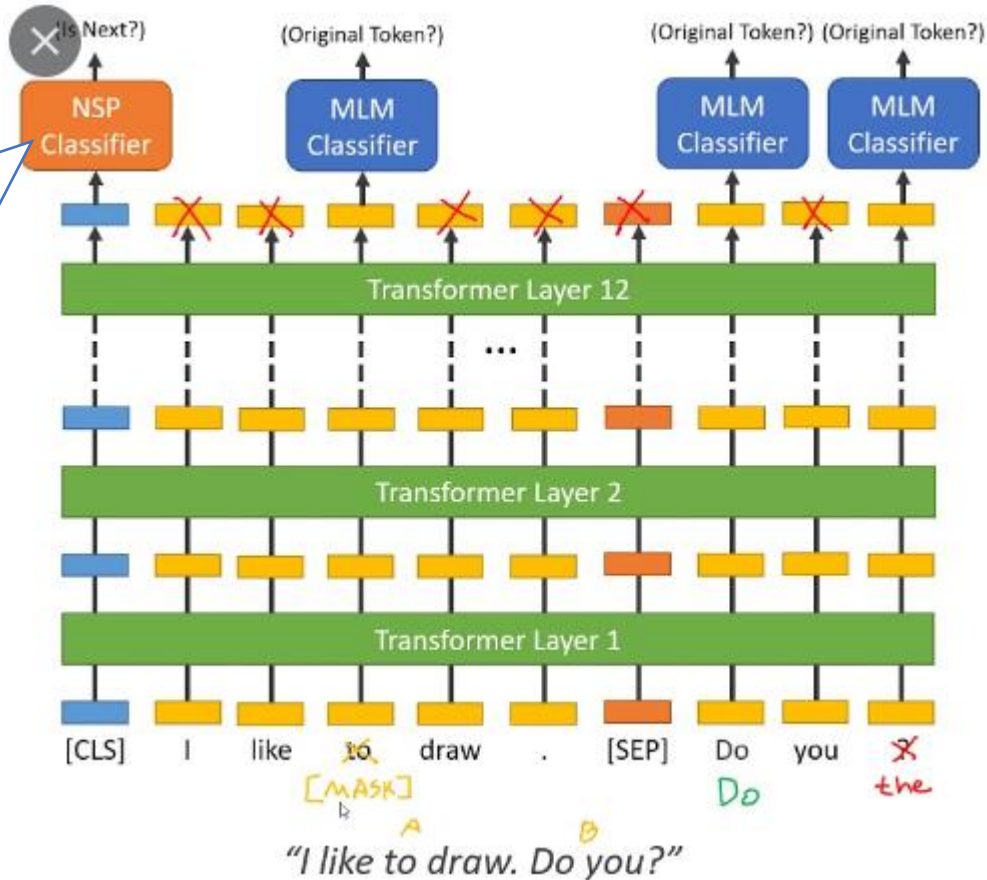
<sup>(2)</sup> Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne, 60205 Compiègne, France

Use Stacked denoising autoencoder (SDA) to resolve the marginal distribution differences between a labeled source domain and an unlabeled target domain.

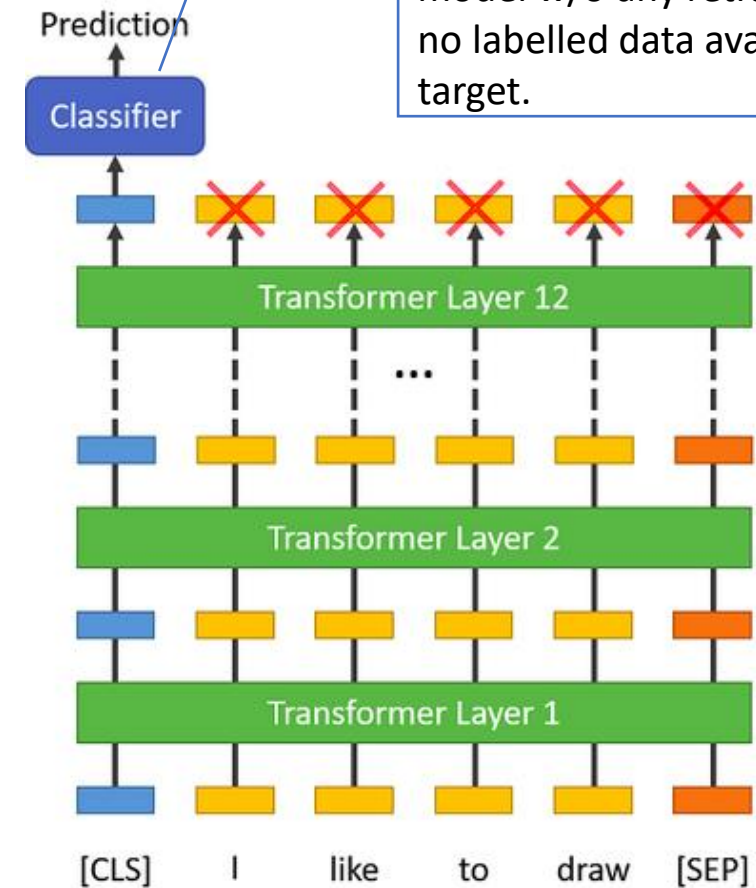
- train SDA with unlabeled data from the source and target domains to transform the input space to discover the common invariant latent feature space.
- train a classifier using the transformed latent features with the labeled source data

# Symmetric Transfer: BERT based pretraining to create doc-embedding

Pretrain with NSP, MLM on both source and target docs to get generic document embedding common for both domains.

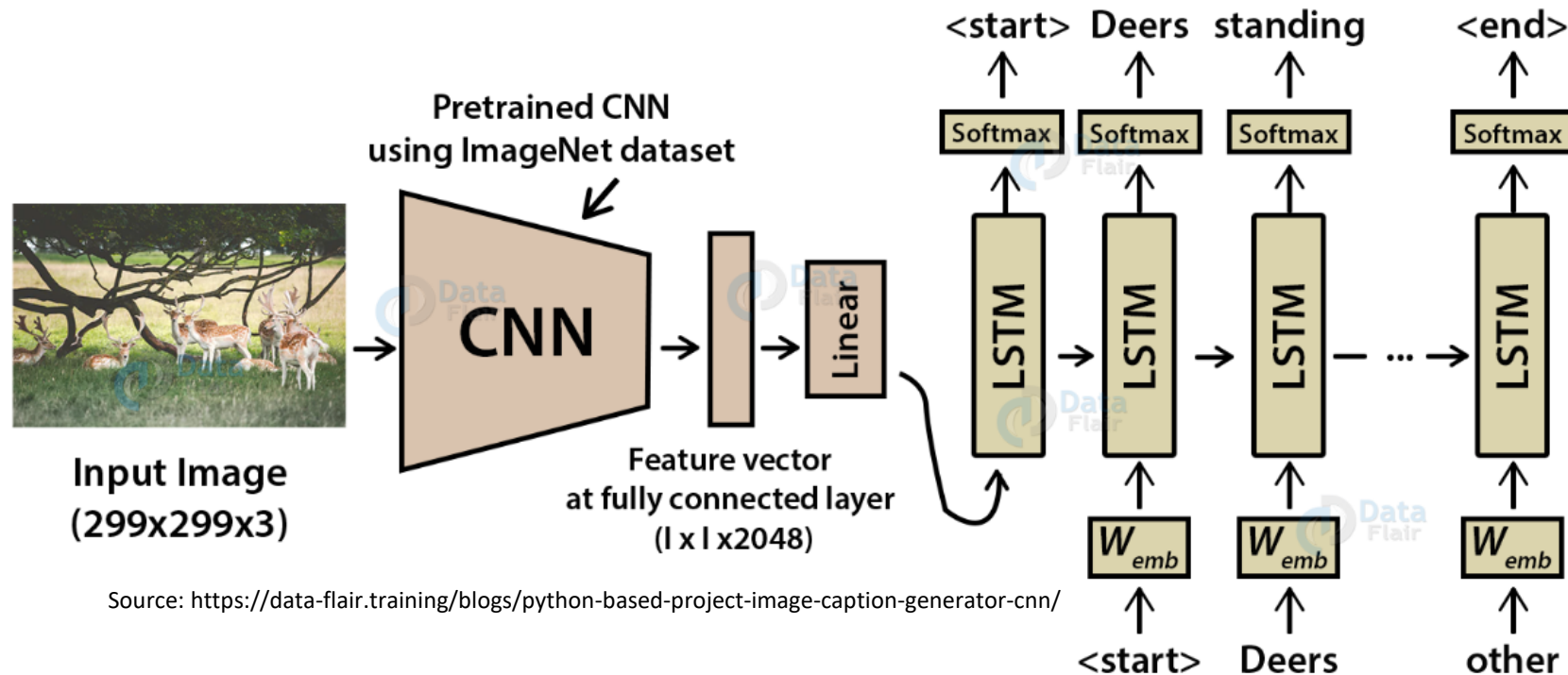


Sentiment Classifier trained on the document embeddings of source model, also performs well for target model w/o any retraining as no labelled data available in target.





# Image Captioning: Symmetric Transfer



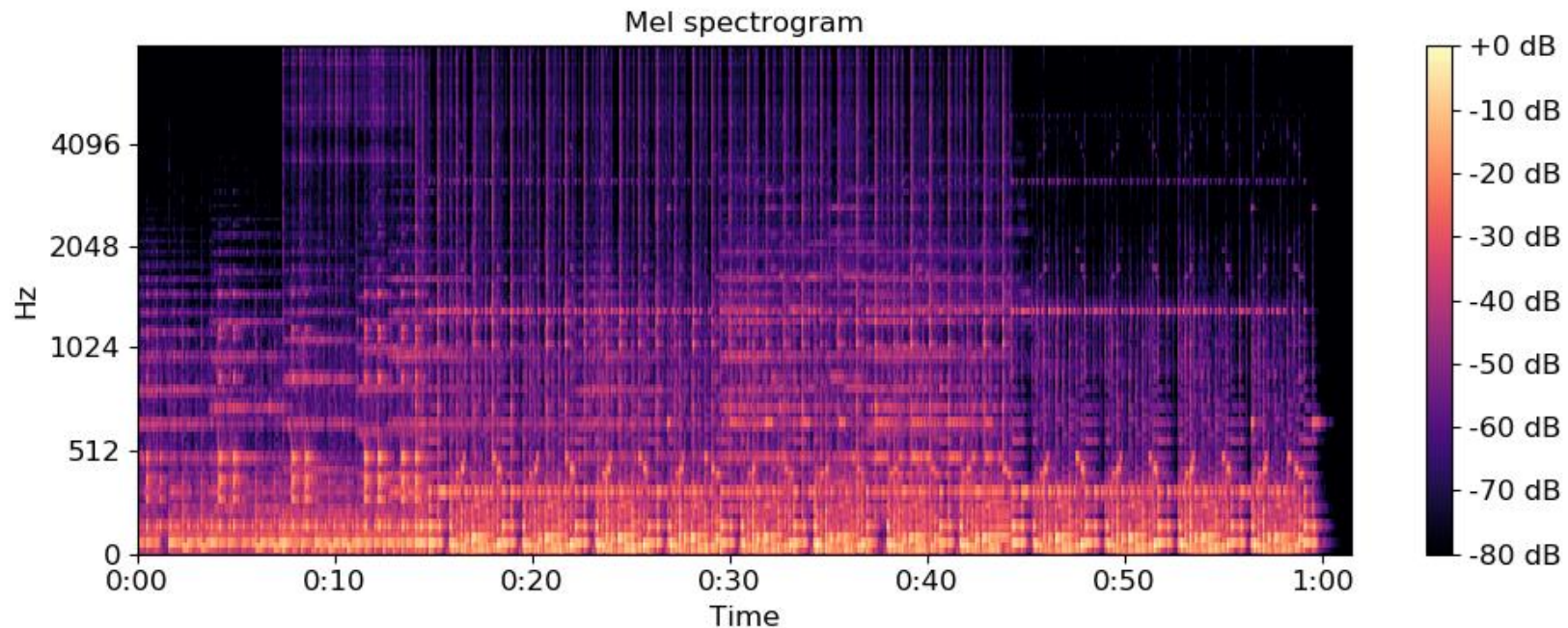
[LINK TO CODE](#)

<https://github.com/tamoghnaG/hands-on-transfer-learning-with-python/tree/master/notebooks/Ch11%20-%20Automated%20Image%20Caption%20Generator>

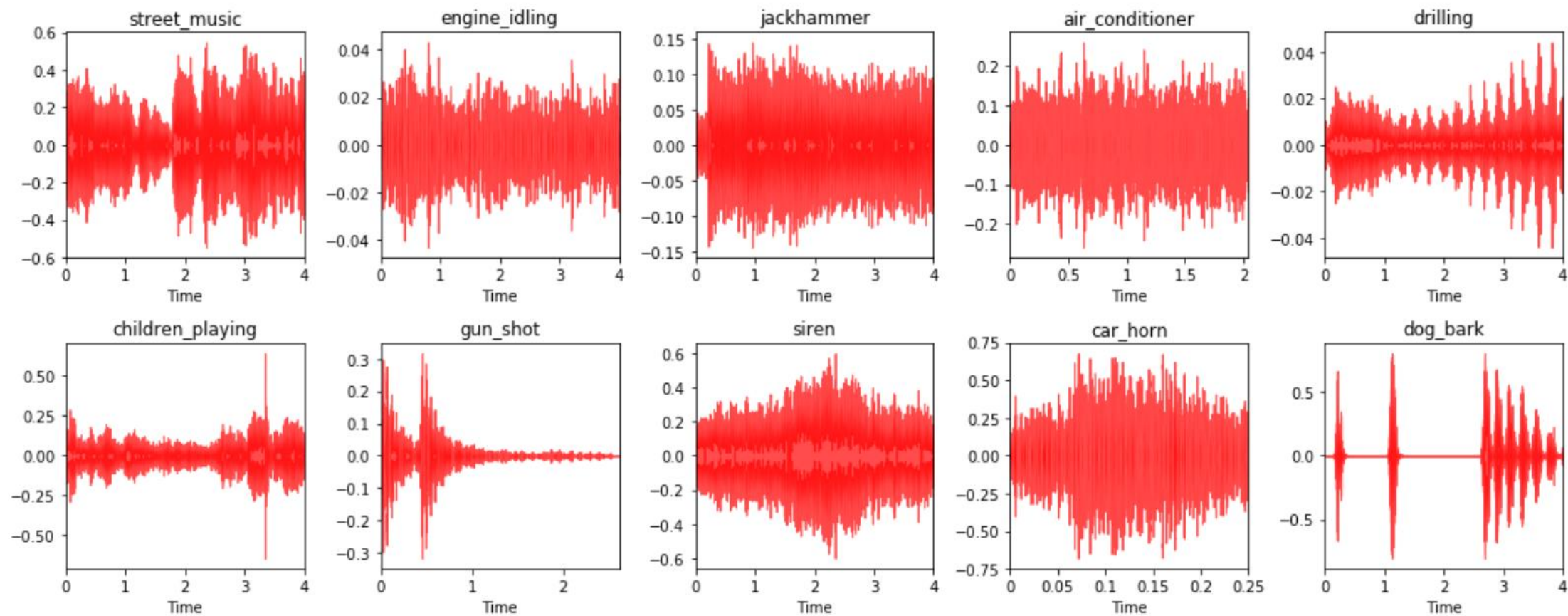
# Module 4: Transfer Learning For Audio Event Detection

# Audio Event Classification

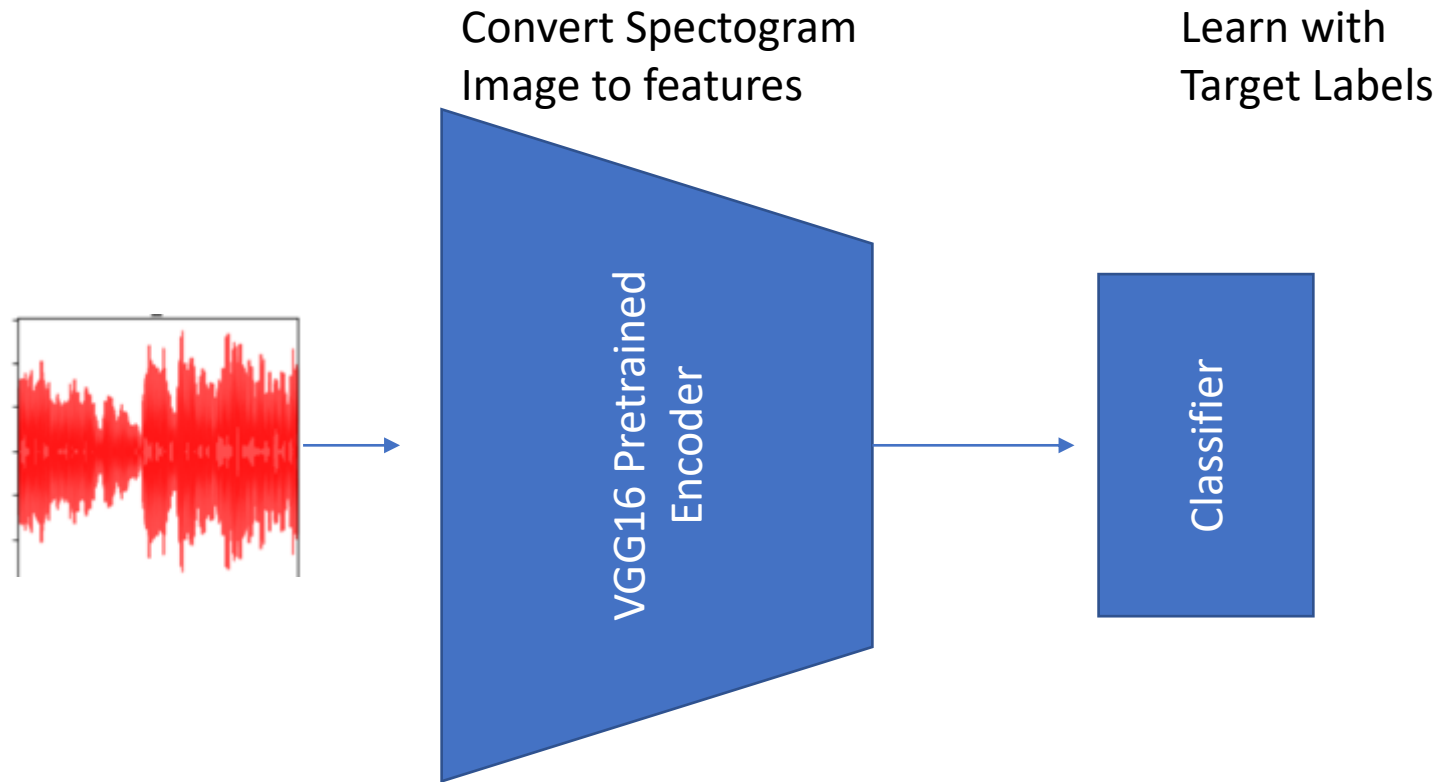
- The UrbanSound dataset consists of 8732 labelled short (less than 4 s) sound excerpts of urban sounds from 10 different classes



# Types of Audio Events



# Model



## LINK to Code:

<https://github.com/dipanjanS/hands-on-transfer-learning-with-python/tree/master/notebooks/Ch08%20-%20Audio%20Identification%20and%20Categorization>

# Module 5: Advanced Applications of Transfer Learning

# Advanced Application of Transfer learning

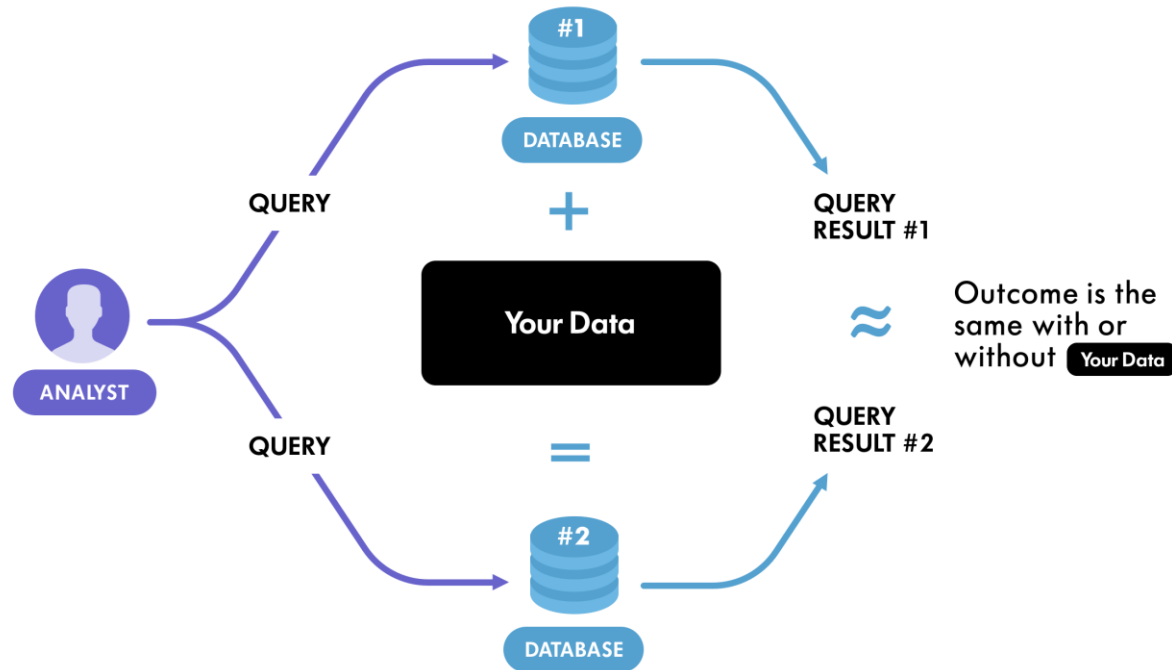
- *Multimodal Deep Learning*: Learn features over multiple modalities (audio, video, text, structured)
- *Privacy Preserving ML*: Train ML models without exposing private data in its original form.

# Privacy Preserving ML

- The goal of privacy-preserving machine learning is to bridge the gap between privacy while receiving the benefits of machine learning.
- Differential Privacy (DP) provides a formal framework for training machine learning models with individual example level privacy.
- Differentially Private Stochastic Gradient Descent (DP-SGD) has emerged as a popular private training algorithm.
  - the computational cost of training large-scale models with DP-SGD is substantially higher than non-private training
- Reference: <https://arxiv.org/abs/2205.02973>



# Differential Privacy



A simple Scenario

Query1: Sum of all for a col M

Query2: Sum of all for a col M

Query1-Query2 = Your Data  
(Privacy Leaked)

Simple Mitigation Strategy:

1. Local: Adding noise before appending to the database.
2. Global: Adding noise before extracting the information

<https://www.section.io/engineering-education/understanding-differential-privacy/>

# Differential Privacy in ML

- We only need loss value to compute gradient
  - So, we can query out private database to return the loss for a data batch. No, need to have direct access to data.
- However, large gradients norms may indicate sensitive data in the selected batch and we may have a privacy breach.

# Overview of DP-SGD

For each example in a minibatch:

1. compute the gradient
2. clip to a pre-selected norm
3. add Gaussian noise before averaging over the minibatch
4. Average over batch and compute SGD step

In practice, DP training can be very expensive or even ineffective for very large models. Not only does the cost of calculating the per-example gradient increase with parameter count, but the norm of the noise added also increases.

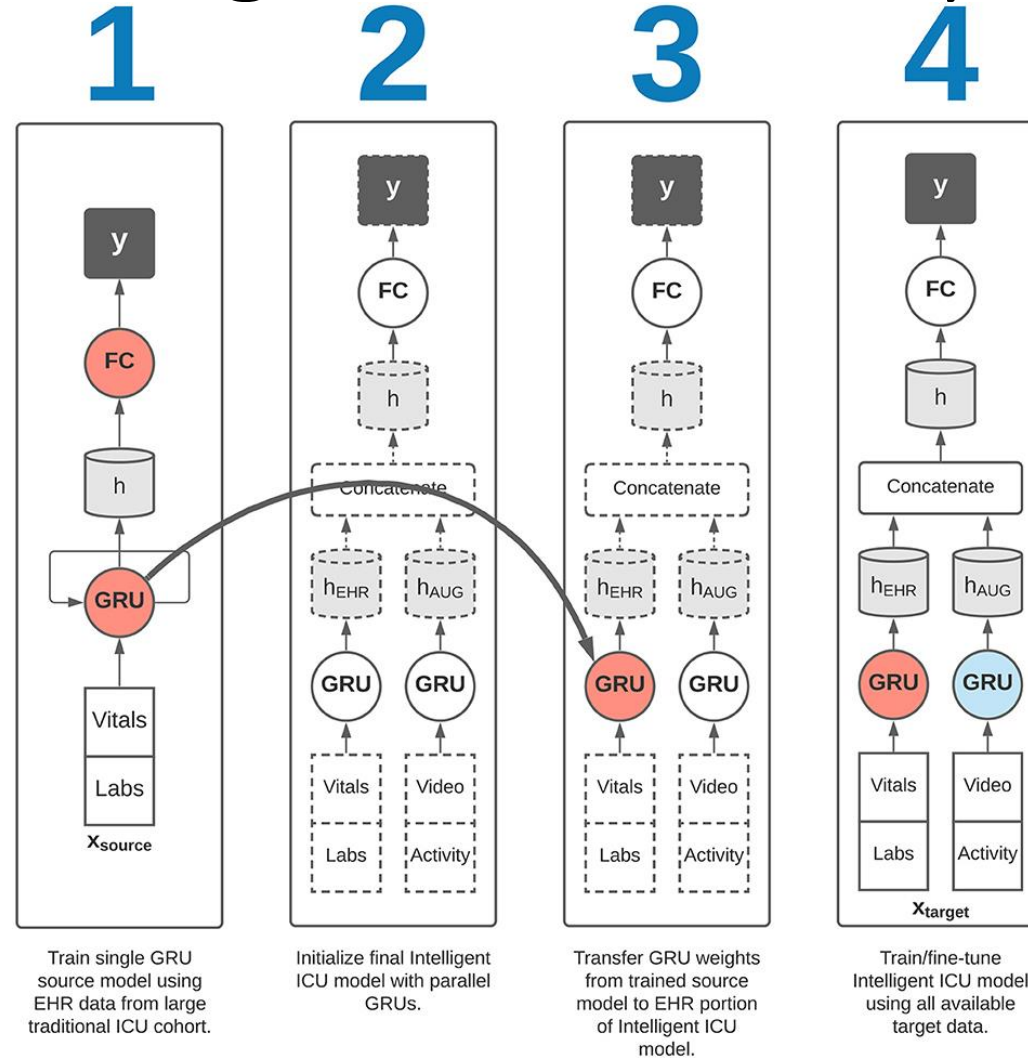
# Transfer Learning For PPML

- Have access to a large public or non-sensitive dataset of the same modality as the private data.
- The model is first pre-trained using the large public or non-sensitive dataset in the traditional non-private way.
- Then at finetuning time do we resort to DP-SGD updates to ensure privacy guarantees for the fine-tuned model.
- Some Recommendations For Implementation:
  1. Increase pretraining data and model size.
  2. Use very large batch sizes in combination with optimizers that can work well with large batch sizes (e.g. LAMB)
  3. Initialize the last layer to zero (or very small) when doing finetuning with DP.

# Multi-Modal Learning

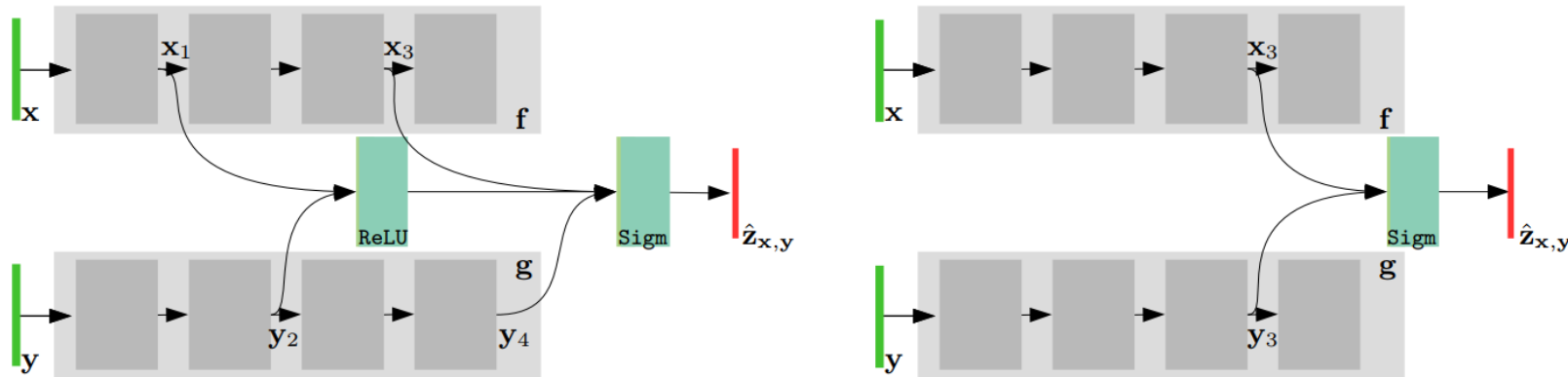
- Data has different modalities
  - Example: A social media post ( video + Text Comments + description)
  - Example: Sensors - Patient Acuity Assessment in the Intelligent ICU ( vital sensors data + patient Video + Audio)

# Transfer Learning: Patient Acuity



# Typical Approach: Multimodal Transfer

- Use pre-trained networks to extract features from each modality and then either concatenate them or project them to a common space.
- Which Layer Of Feature Extractor to use? Brute-Force?



# MFAS: Multimodal Fusion Architecture Search

- Exhaustively exploring all possible architectures is intractable.
- A surrogate function is trained during progressive exploration of the search space, and it is used to reduce the amount of neural networks that have to be trained and evaluated by predicting performance of unseen architectures
- Reference: <https://arxiv.org/abs/1903.06496> by JM Pérez-Rúa · 2019 · Cited by 117



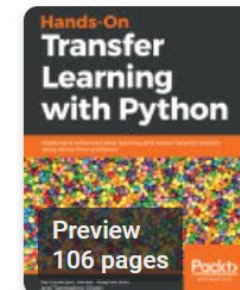
# Conclusion

- We learned about various types of transfer learning
- Deep Transfer Learning: Domain Adaptation & Domain Confusion
- Advanced Transfer Learning Applications
- More on Transfer Learning
- <https://www.amazon.in/Hands-Transfer-Learning-Python-TensorFlow/dp/1788831306>

## Hands-On Transfer Learning with Python

Implement Advanced Deep Learning and Neural Network  
Models Using TensorFlow and Keras

By [Dipanjan Sarkar](#), [Raghav Bali](#), [Tamoghna Ghosh](#) · 2018



BACKUP

# Multi Model Knowledge Transfer (MMKT)

- Learning Categories from Few Examples
  - A lot of annotated data on a source problem and the need to solve a different target problem with few labeled samples, where source and target present a distribution mismatch
  - Example: We want to learn the target object class *okapi* from few examples, having already a model for the source categories *horse*, *zebra*, *melon* and *apple*.
    - The okapi looks like horse / zebra. Thus, in the learning process we would like to transfer information from these two categories.



okapi

# How to Transfer Parameters

- Train Binary Classifiers on each of the source classes ( One-vs-Rest)
- A Linear classifier will look like the following:
  - $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$
  - Here  $\phi(\mathbf{x})$  is a feature mapping, that maps the samples into a high, possible infinite dimensional space, where the dot product is expressed with a functional form  $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$  - the kernel function
- Given  $J$  source classifiers for each source:  $f_j(\mathbf{x}) = \hat{\mathbf{w}}_j^T \phi(\mathbf{x}) + b$
- The target classifier parameters  $\mathbf{w}$  can be defined as a weighed combination of the source parameters :  $\mathbf{w} = \sum_j \beta_j \hat{\mathbf{w}}_j$
- From few target labels we can learn the  $\beta_j$ s.

# Finding the domain weights $\beta_j$

- The transfer learning problem boils down to finding the weights  $\beta_j$  such that we can find the target classifier weights as  $\mathbf{w} = \sum_j \beta_j \hat{\mathbf{w}}_j$
- This can be formulated as an LS-SVM (least square support vector machine) type optimization problem.
- In LS-SVM, due to the use of equality constraints in the formulation of optimization problem, a set of *linear equations* must be solved instead of a quadratic programming problem in classical SVM

Adaptive LS-SVM Formulation for finding weights  $\beta_j$  from limited (N) target labelled data

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \left\| w - \sum_{j=1}^J \beta_j \hat{w}_j \right\|^2 + \frac{C}{2} \sum_{i=1}^N \zeta_i \xi_i^2 \\ \text{s.t.} \quad & y_i = w^\top \phi(x_i) + b + \xi_i, \quad \forall i = 1, \dots, N, \end{aligned}$$

parameter  $\zeta_i$  can be used to balance the contribution of positive and negative samples

$$\zeta_i = \begin{cases} \frac{N}{2N^+} & \text{if } y_i = +1 \\ \frac{N}{2N^-} & \text{if } y_i = -1 \end{cases} . \quad (9)$$

Here  $N^+$  and  $N^-$  represent the number of positive and negative examples respectively.