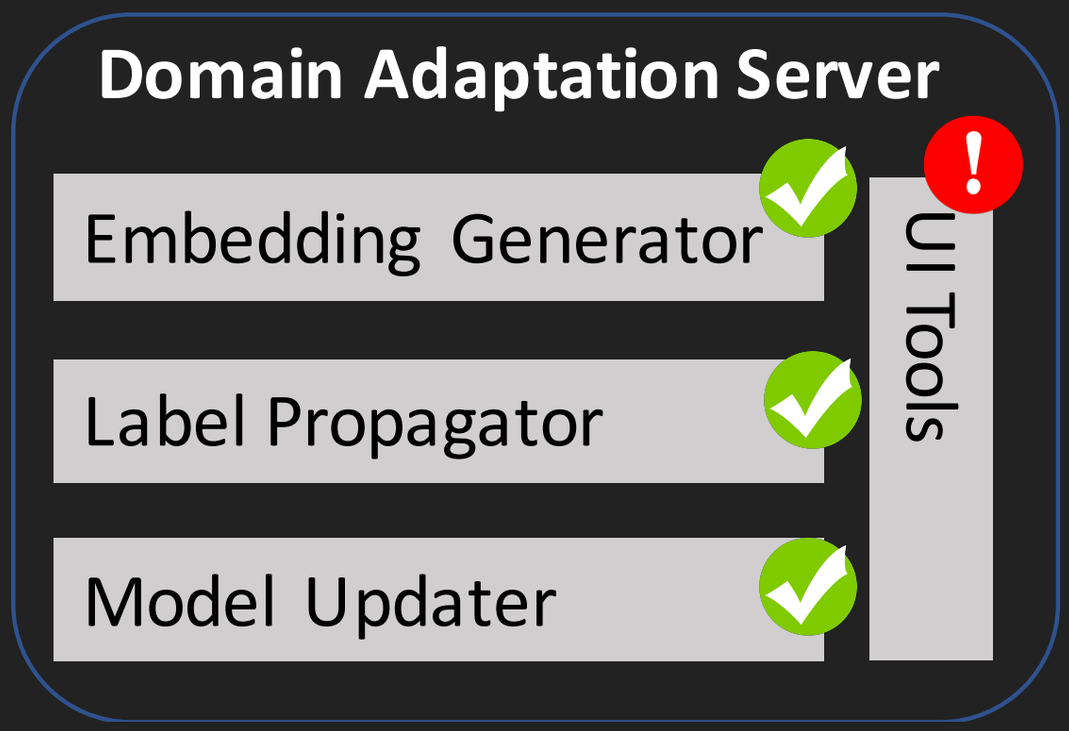
JCI INNOVATION GARAGE

DOMAIN ADAPTATION BACKEN

DOMAIN ADAPTATION–ENGINEERING TOOL

- Demonstration from a engineering tool point of view
- For an end-user, we will need to freeze some inputs, but the backend remains the same

- https://github.com/ashoksundaresan/online-learning



TEXT

Johnson Controls

Deep Learning Model Domain Adaptation Tool

Connect to Gateway

Gateway Address

Data endpoint

Username

Password

Camera ID

Model ID

CONNECT

Status: Connected to <>

Detected: 300 Images

Fetched:

0

FETCH IMAGES

TEXT

Johnson Controls

Deep Learning Model Domain Adaptation Tool

Connect to Gateway

Gateway Address

Data endpoint

Username

Password

Camera ID

Model ID

Backend complete

Placeholder exists, minor coding needed

Does not exist

CONNECT

Status: Connected to <>

Detected: 300 Images

Fetched:

0

FETCH IMAGES

TEXT

Johnson Controls

Deep Learning Model Domain Adaptation Tool

Generate Embeddings

Base Model

Model Crop Size

Performance Layers

Prediction Key

LOAD

Target Data File

Embeddings Layer

ML <ID> Loded

MODEL ARCHITECTURE: /HOME/

MODEL LAYERS: L

DEPLOY WEIGHTS:/……../

SELECT OTHER WEIGHTS: W

Source data needs to be inferred again!

COMPUTE

0

TEXT

Johnson Controls

Deep Learning Model Domain Adaptation Tool

Generate Embeddings

Base Model

Model Crop Size

Performance Layers

Prediction Key

LOAD

Target Data File

Embeddings Layer

Destination Folder

ML <ID> Loded

MODEL ARCHITECTURE: /HOME/

MODEL LAYERS: L

DEPLOY WEIGHTS:/……../

SELECT OTHER WEIGHTS: W

Source data needs to be inferred again!

COMPUTE

0

TEXT

Johnson Controls

Deep Learning Model Domain Adaptation Tool

Determine Labels

Input Folder

Destination Folder

FEATURE SPACE

LABEL EXTRACTION METHOD

	LABEL SPREADING	RF	SVM	GT	DL	VOTING
Embeddings	x	x	x			x
TSNE	x	x	x			
PCA	x	x	x			
Embeddings + Autoencoders						
Embeddings + Autoencoders+TSNE						
Embeddings + Autoencoders+PCA						
Induction Learning Based (Multiple autonecoders)						

TEXT

Johnson Controls

Deep Learning Model Domain Adaptation Tool

Determine Labels

Input Folder

Destination Folder

FEATURE SPACE

LABEL EXTRACTION METHOD

	LABEL SPREADING	RF	SVM	GT	DL	VOTING
Embeddings	x	x	x			x
TSNE						
PCA	x	x	x			
Embeddings + Autoencoders !						
Embeddings + Autoencoders+TSNE !						
Embeddings + Autoencoders+PCA !						
Induction Learning Based (Multiple autonecoders) !						

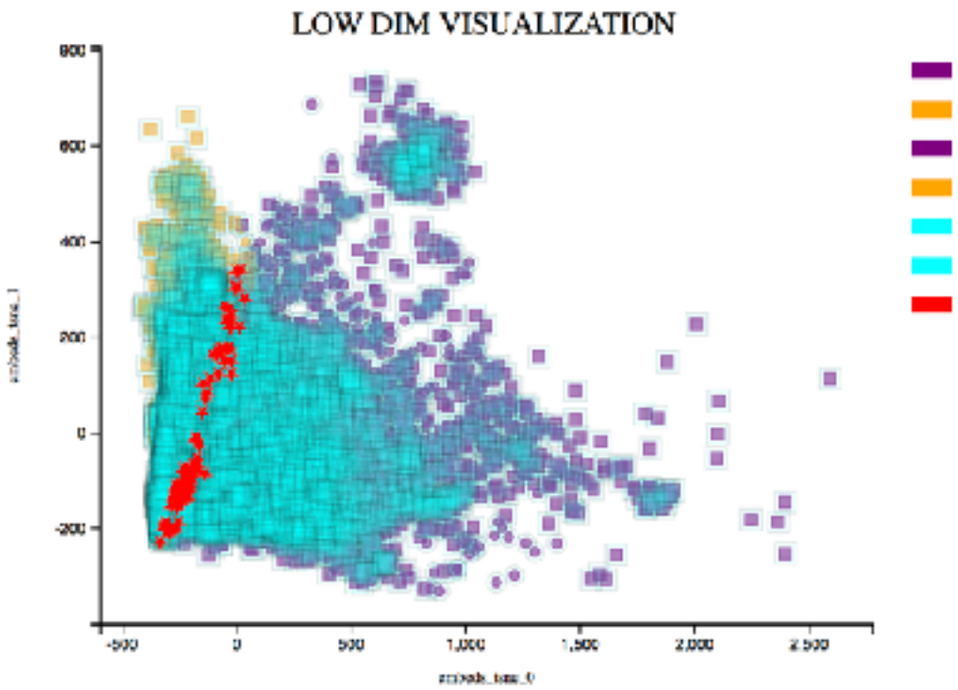
TEXT

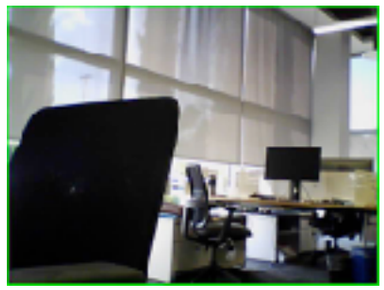
Johnson Controls

Deep Learning Model Domain Adaptation Tool

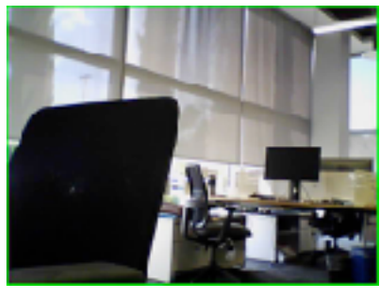
Validate Labels

Method

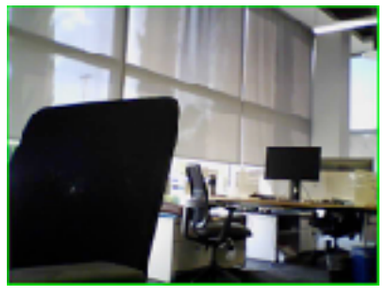




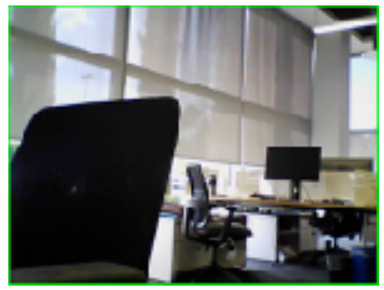
Disagree



Disagree



Disagree



Disagree

SAVE LABELS

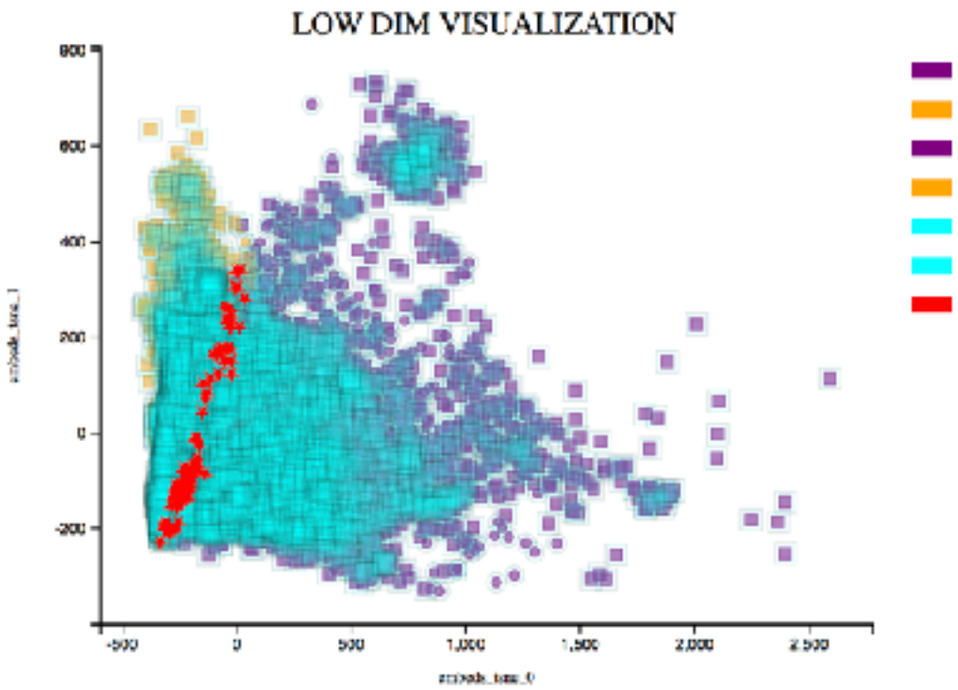
TEXT

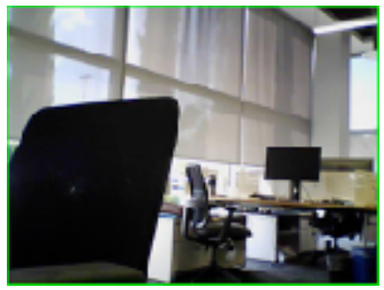
Johnson Controls

Deep Learning Model Domain Adaptation Tool

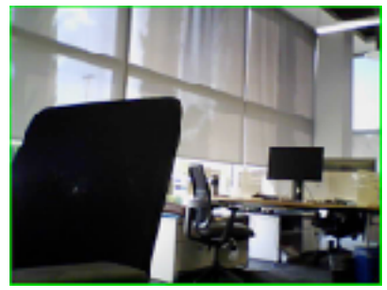
Validate Labels

Method

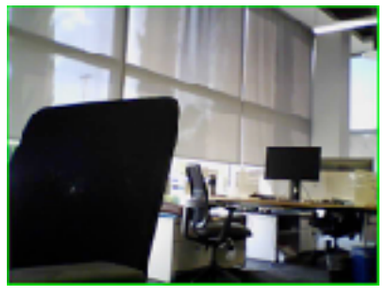




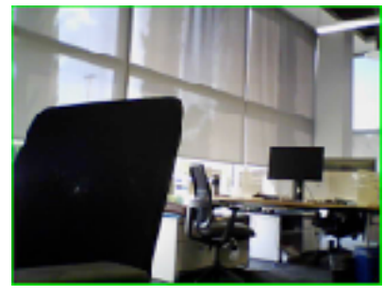
Disagree



Disagree



Disagree



Disagree

SAVE LABELS

TEXT

Johnson Controls

Deep Learning Model Domain Adaptation Tool

Retrain

Base Model

Model Crop Size

Performance Layers

Prediction Key

LOAD

Target Data File

Embeddings Layer

ML <ID> Loded

MODEL ARCHITECTURE: /HOME/

MODEL LAYERS: L

DEPLOY WEIGHTS:/……../

SELECT INITIAL WEIGHTS: W

TRAIN

0

TEXT

DESCRIPTION (WORK IN PROGRESS): HIGH LEVEL

Load_trainer

Inputs
dir_path: Path to where a model (saved by ML_Train) is saved.

Attributes:
dir_strc: object with Location of components of a trainer.
available_weight_files:.
deploy_weight_file:.
final_weights_file:.

Methods:
set_deploy_weights_file(one_of_the_available_weights_file)

get_performance(data_file,crop_size,prediction_key,save_data_flag=True,labels_col=[],layer='',weights_file='',save_data_suffix='')--> **Computes predictions and embeddings and confusion matrix (if label_col is present), save them to analysis_results_files, with the same tag as the name in data_file and a user provided suffix. Returns: Prediction data frame and confusion matrix array**

Embeds_classify

Inputs
source_embds_file,target_embds_file,source_data_txt_file,target_data_txt_file
embds_param={'type':'convs','n_convs':2,'standardize':False}true_labels_col='1'
index_col='',sep=','
source_test_frac=.2
target_test_frac=2
serving_dir=''
inter_save=True
dl_pred_key='dl_pred'

Methods:
set_test_train_data(source_test_mask=[],target_test_mask=[],source_test_frac=.2,target_test_frac=.2)

--- Dimensionality reduction
compute_tsne(perplexity=30, n_components=2, init='pca',n_iter=3500,save_data_viz=True,train_feat_space='embeds')

compute_pca(self,train_feat_space='embeds')

dim_red_autoencode(encode_dims=[128,64])(Note: currently commented out because no tensor flow on instance, tested on local machine)
---- Label generation
low_dim_classifier(methods=['RandomForestClassifier'],train_feat_space='embeds',train_mode='source_and_target')

source_to_target_label_prop(train_feat_space='embeds',kernel_param={'type':'rbf','gamma':20})

---- Samples to suggest for manual labeling
get_samples_tolabel(self,dl_prob_label)

save_perforamance(self,save_dir,class_key='all',print_flag=True,suffix='')

Domain_adaptation

Inputs
source_files,source_labels,target_files,base_model_files,params,name_prefix,serving_path,target_labels=[]
params = {'mix_pct': [(0, 99)], 'crop_size': 227,'perf_layers': ['loss'],'prediction_key': 'prob','source_test_train_split_frac':.2,target_test_train_split_frac':.2}

Create multiple machine learning models to train based on ML_trainers based on number of elements in mix_pct, and modifying the base_model_files

Methods:
split_test_train()
setup_training_models()
generate_training_val_data()
train_transfer_learning(solver_method='pycaffe')
get_performance()

DESCRIPTION (WORK IN PROGRESS): MIDDLE LEVEL

include.ML_Train

Inputs
model_id,serving_dir='',caffe_root
defaults(Model_timestamp,cur_dir,os.environ["CAFFE_ROOT"])
Init creates: directory structure(dir_strc), logger
Note there are two loggers:
a) for logging the parameters /files for training
b) For logging the training process (in dir_strc.weight_files

Methods:
---Create data
create_img_lmdb_os(source,resize_height ,resize_width,out_base_name=[])
create_img_lmdb(source,img_width,img_height,img_channels,sep=' ',out_base_name=[])
compute_mean(lmdb_source,mean_file_path=[])
set_lmdbs(train_lmdb, test_lmdb,mean_file,crop_size,img_channels)

---- Generate prototxt files from present topology (only Alexnet supported currently))
gen_archProtoFiles(topology, params, run_tag='base',batch_size=128)
gen_solverProtoFiles(train_proto_path,solver_params,test_proto_path='',run_tag=[])

---- Create prototext files from existing files
set_archProtoFiles(train_proto_in,test_proto_in,deploy_proto_in, run_tag='base')

----
train_solver(solver_proto,niter=1000,weights_file=[],perf_layers=[],display_interval=[],weights_save_interval=[],run_tag=[])

include.Low_dim_classifier

Inputs
x,y,methods=['SVC']
Supported methods: [KNeighborsClassifier, SVC_Linear, SVC, GaussianProcessClassifier, DecisionTreeClassifier,RandomForestClassifier,MLD_CLassifier,AdaBoostClassifier,GaussianNB,QuadraticDiscriminantAnalysis]
'all' create an ensamble with all classifiers above

Methods/Attributes:
classifiers: List of available classifier
train(train_methods=['SVC'])
predict(x_test,test_methods=['SVC'])
get_performance(x_test,y_test,perf_methods=['SVC'])

To add:
get_ensamble_prediction()[currently these are computed outside, at the higher level]
get_ensamble_performance()

include.autoencode_reps

Inputs

reps_to_learn: Representations to learn
reps_to_process=np.array([])
encode_dims=[128,64,32]# Dimensions of hidden layers
params={}# Empty implies, defaults, see code

return low_dim_embds_learn: shape:
[len(reps_to_learn],encode_dim[-1]]

If available:
low_dim_embds_processed_reps
[len(reps_to_process],encode_dim[-1]]

Notes:
Checkpoints created in './log_dir'
params:
optimizer
Loss
batch_size
regularizer_type: 'l1' or None
regularizer_val: sets the sparcity if 'l1'

include.four_group_plot

four_group_plot(csv_file,x_col,y_col,group1_col,group2_col,group3_col,show_only_col,show_only_col_flag=True)

Show selectable plot in browser.
Can return html/d3 object

TEXT

DESCRIPTION (WORK IN PROGRESS): LOW LEVEL

include.Inference

include.train_solver

\

include.mix_source_target

include.sample_to_label