Automatic detection of rapid changes of land cover (snow) from temporal InSAR coherences using machine learning

Smith Pataraprasitpon

Table of Contents

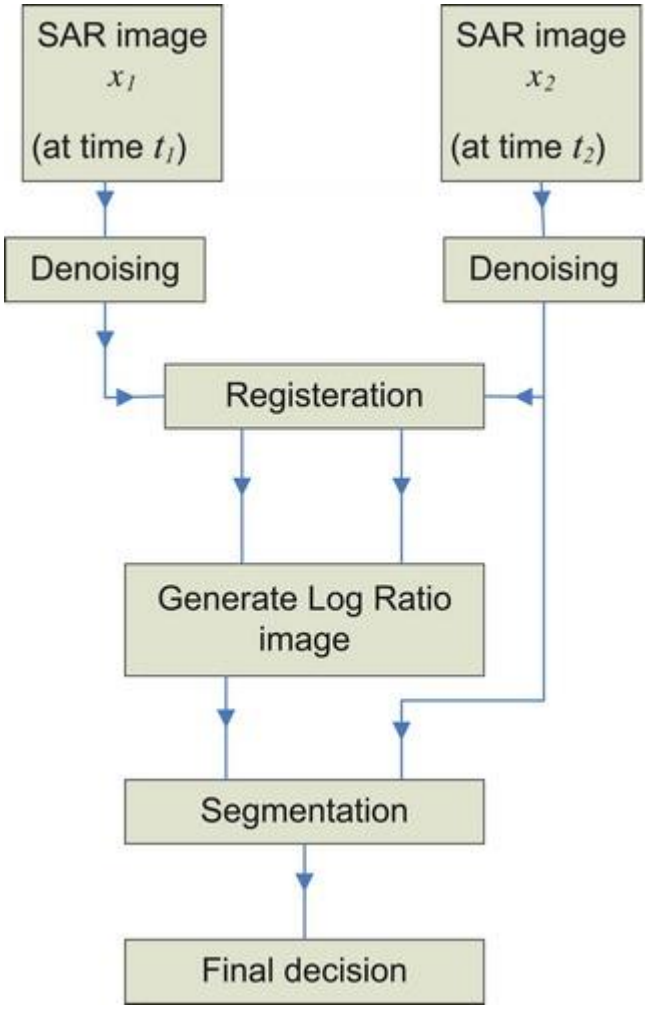
Introduction - Change Detection in Satellite Images

- One common type of multitemporal analysis is change detection.
- Change detection involves the direct comparison of two or more images to identify how areas change over time.
- Various methods of change detection are possible.

Change detection has been widely used to assess:

- Shifting cultivation.
- Deforestation.
- Urban growth.
- Impact of natural disasters like tsunamis, earthquakes, and use/land cover changes.

Introduction - Change Detection in Satellite Images



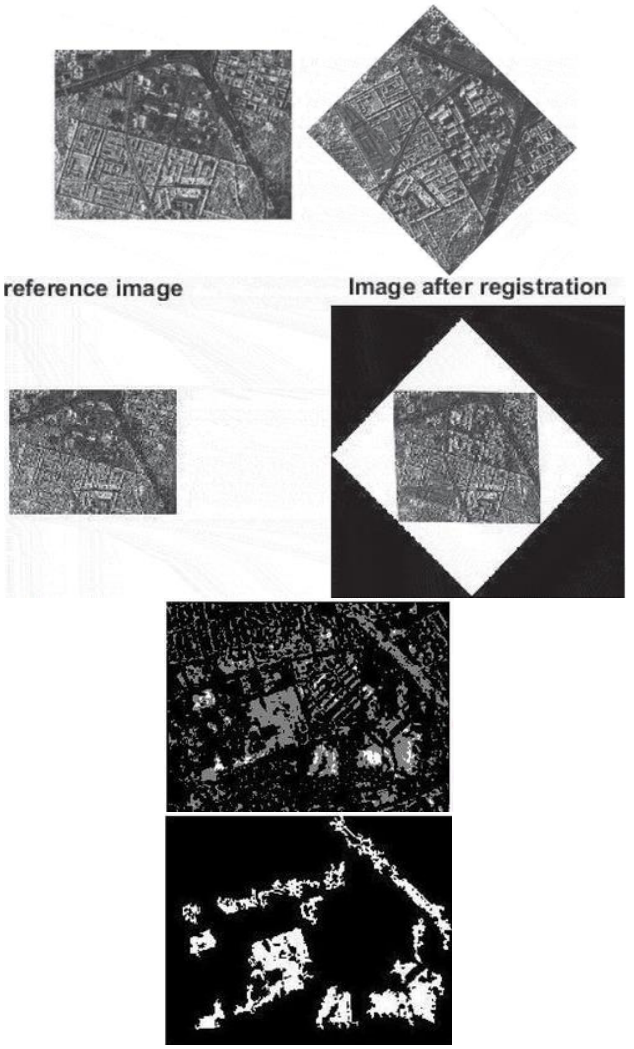


Fig. 1. Registration-based change detection for SAR images. (Alshimaa Y et al.: REGISTRATION-BASED CHANGE DETECTION FOR SAR IMAGES)

Introduction - Interferometric coherence for land classification using machine learning

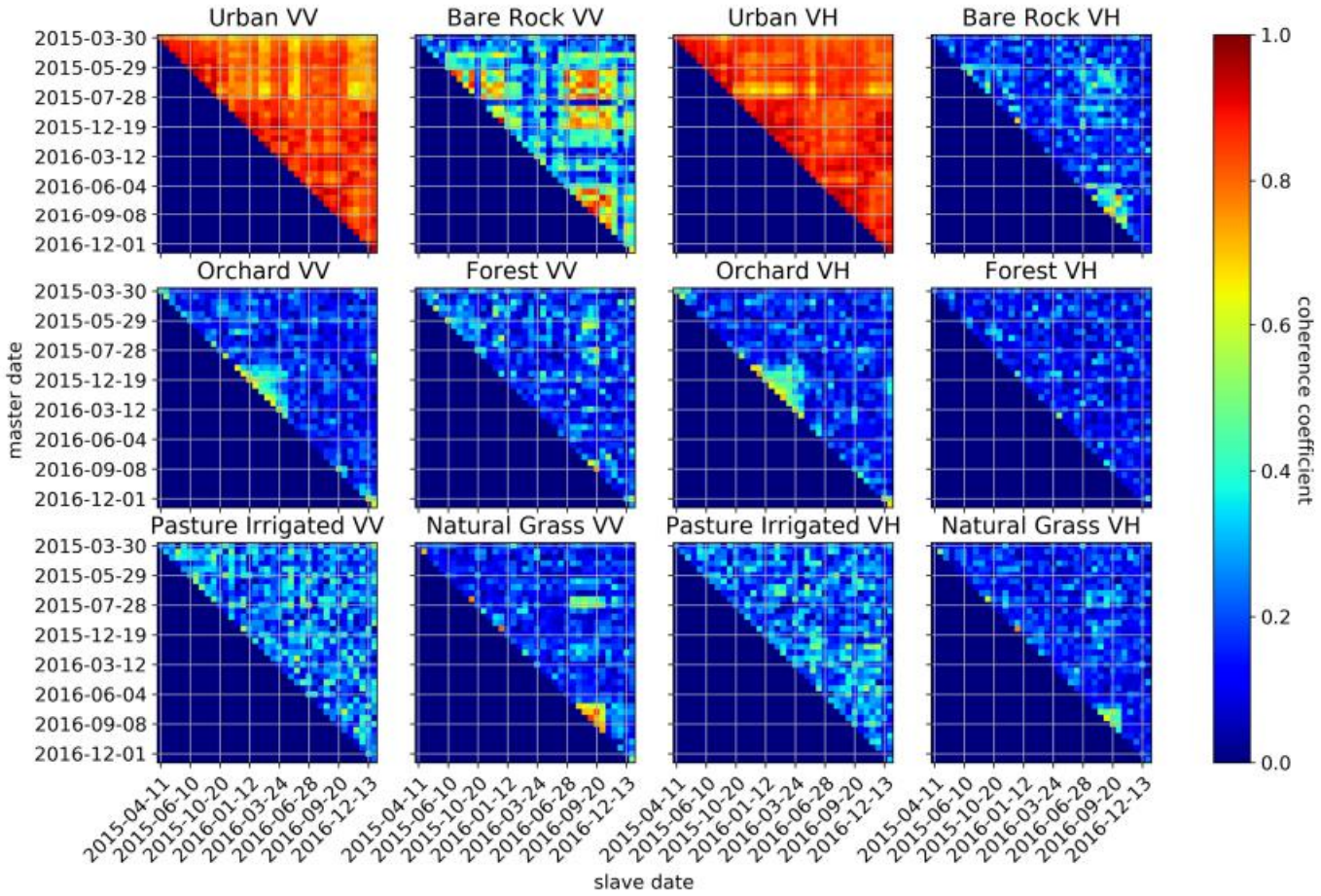


Fig. 2. Comparison of the dynamics of multitemporal coherence matrices in six different land cover types (urban, bare rock, orchards, forest, pasture irrigated, natural grassland) containing all possible combinations of temporal baselines for the six selected pixels of the six different land cover types. (JACOB et al.: SENTINEL-1 INSAR COHERENCE FOR LAND COVER MAPPING)

Objective - Constructing machine learning pipeline for snow detection

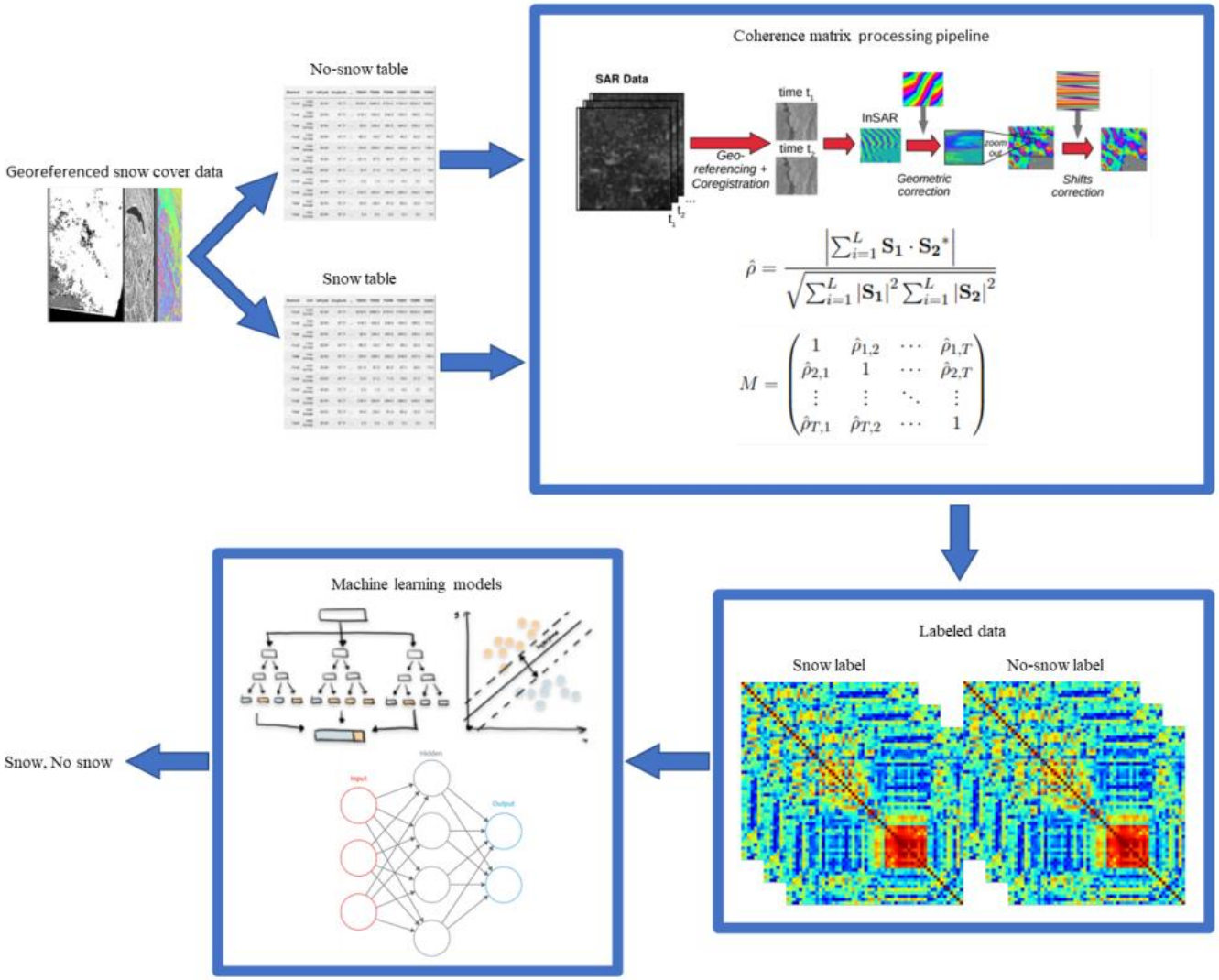


Fig. 3. Machine learning pipeline for snow detection.

Scope - Satellite Data, Study Area

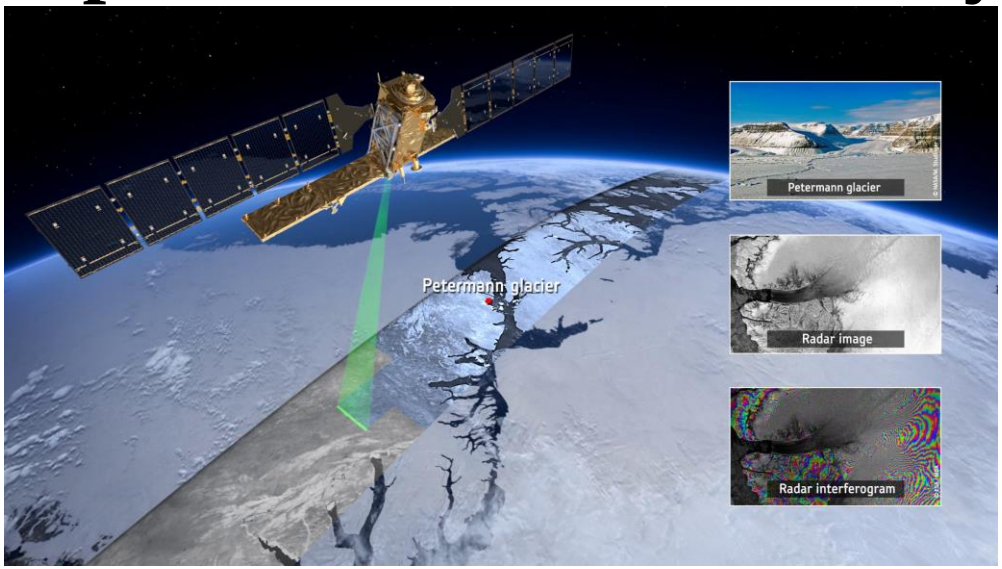


Fig. 4. Sentinel-1. (Figure from https://www.esa.int/)

Fig. 5. Sentinel-2. (Figure from https://www.esa.int/)

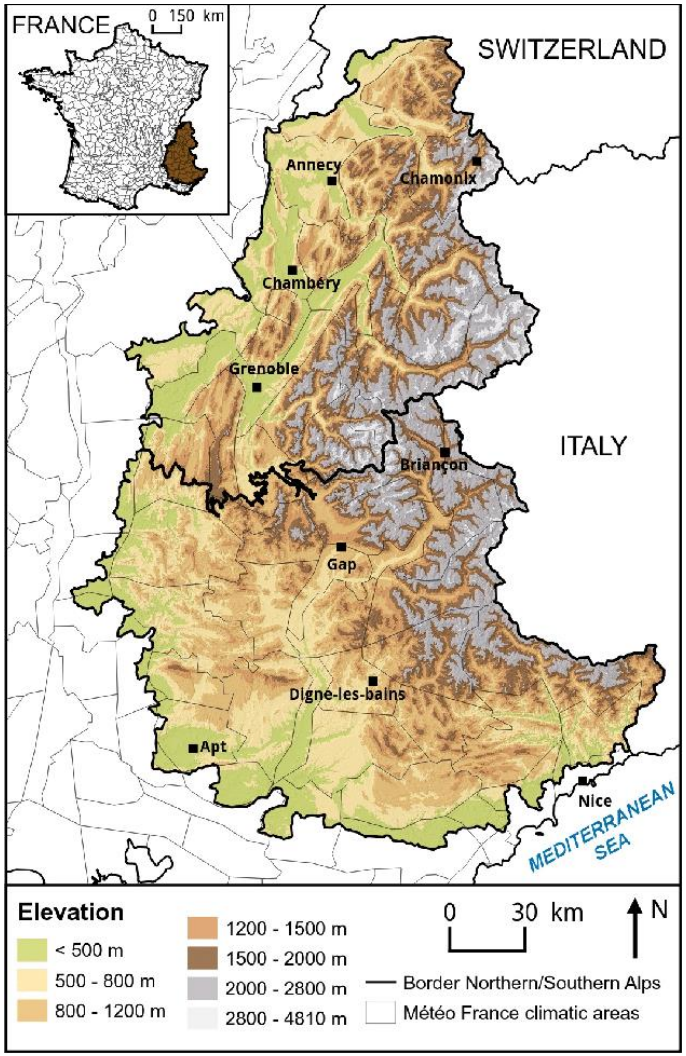


Fig. 6. French Alps. (Figure from https://www.researchgate.net/)

Theoretical background - Snow Cover Using Spaceborne SAR

- SAR is often preferred over optical imagers for these applications because its performance is independent of available daylight and visibility.

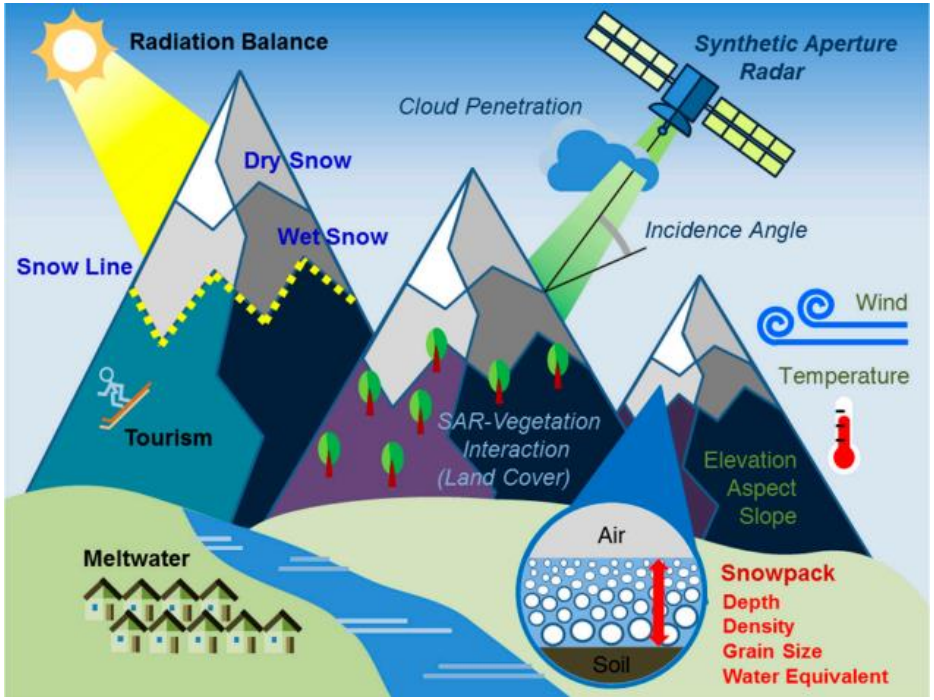


Fig. 7. Illustration of different snow types and snow line (deep blue font), the importance of snow (black font), synthetic aperture radar (SAR)-related characteristics (font in italics), factors influence snow (green font) and snowpack parameters (red font). (S. Tsai et al.: REMOTE SENSING OF SNOW COVER USING SPACEBORNE SAR: A REVIEW)

Theoretical background - SAR Data Preprocessing

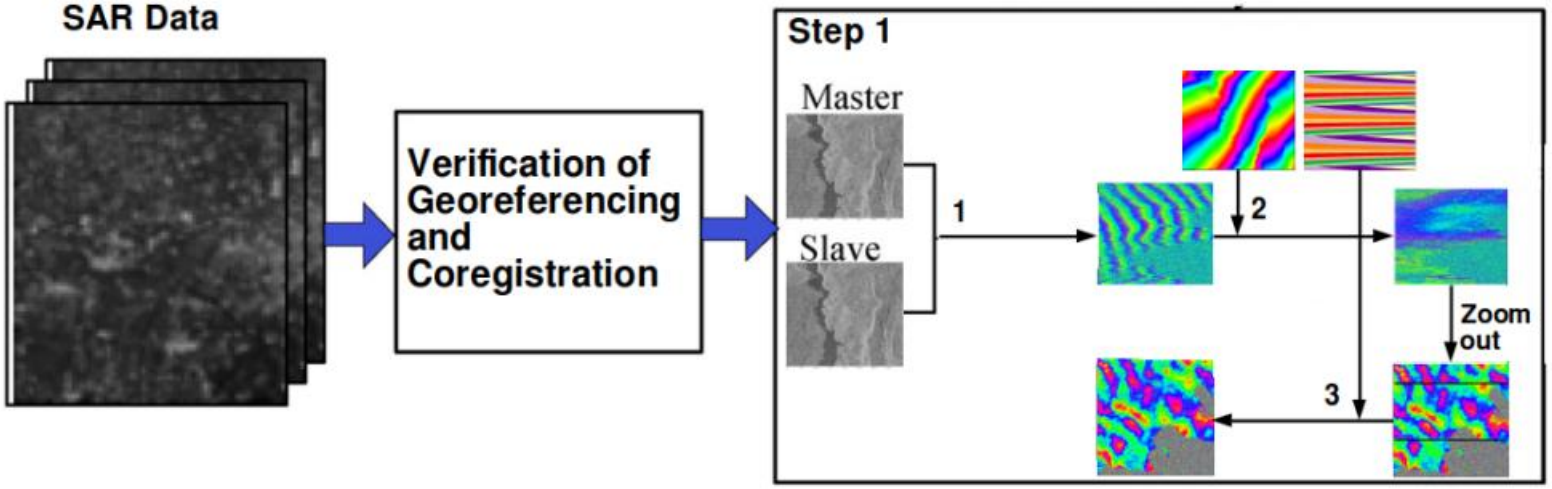


Fig. 8. The processes from complex (pure) SAR images through the building of an interferogram and the adjustments of its various biases to generate InSAR data. (Figure from Salah Eddine Boudaour)

1. *Element-wise (or Hadamard) Product on the 2 complex Master and Slave complex images is done by*

$$\xi^{\circ} = S_1 \bullet S_2^*$$

2. *Geometrical correction is done by*

$$\xi^{*} = \xi^{\circ} \exp(j\Phi_{Geom})$$

3. *Persistent phase-shifts correction is done by*

$$\xi^{\surd} = \xi^{*} \exp(-j\phi_{Shifts})$$

Theoretical background - Temporal coherence

The complex correlation coefficient can be written as

$$C = e^{i\phi} \cdot \rho = \frac{E(S_1 \cdot S_2^*)}{\sqrt{E(|S_1|^2)E(|S_2|^2)}}$$

For each pixel can be substituted by the averaging of its neighbors, and the estimator is

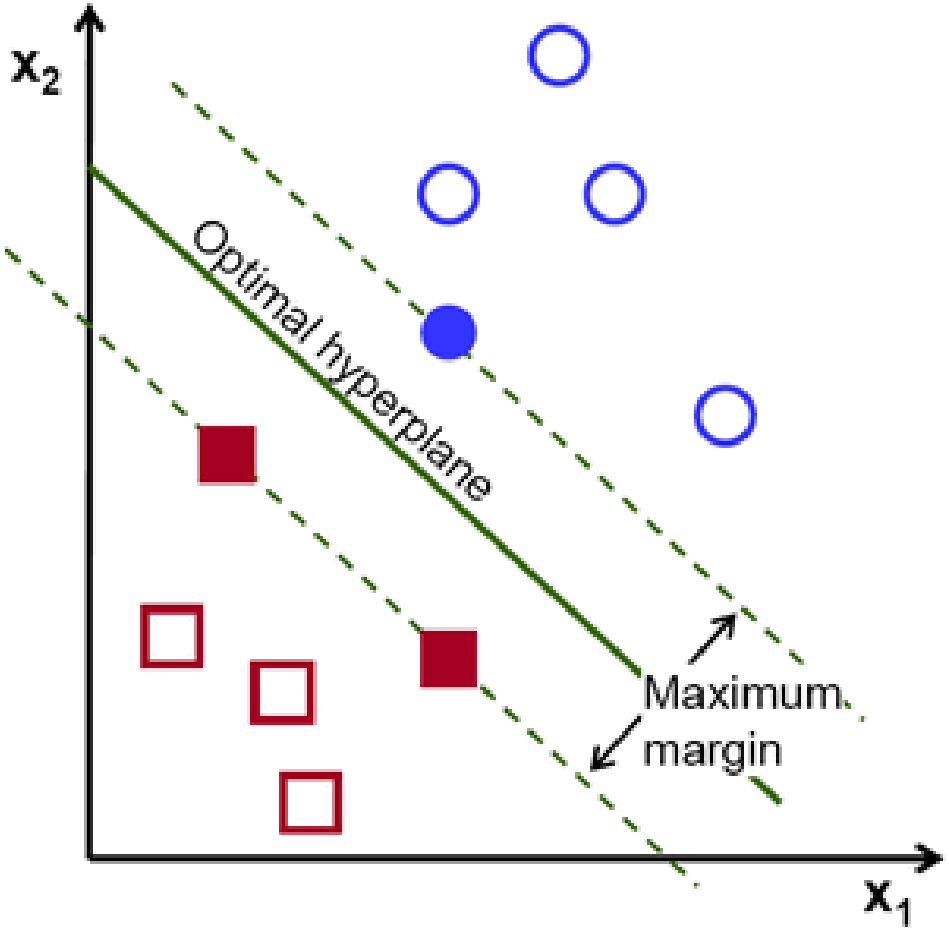
$$\hat{\rho} = \frac{|\sum_{i=1}^{L} S_1 \cdot S_2^*|}{\sqrt{\sum_{i=1}^{L} |S_1|^2 \sum_{i=1}^{L} |S_2|^2}}$$

Theoretical background - Temporal Coherence Matrix representation

$$M = \begin{pmatrix} 1 & \hat{\rho}_{1,2} & \cdots & \hat{\rho}_{1,T} \\ \hat{\rho}_{2,1} & 1 & \cdots & \hat{\rho}_{2,T} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}_{T,1} & \hat{\rho}_{T,2} & \cdots & 1 \end{pmatrix}$$

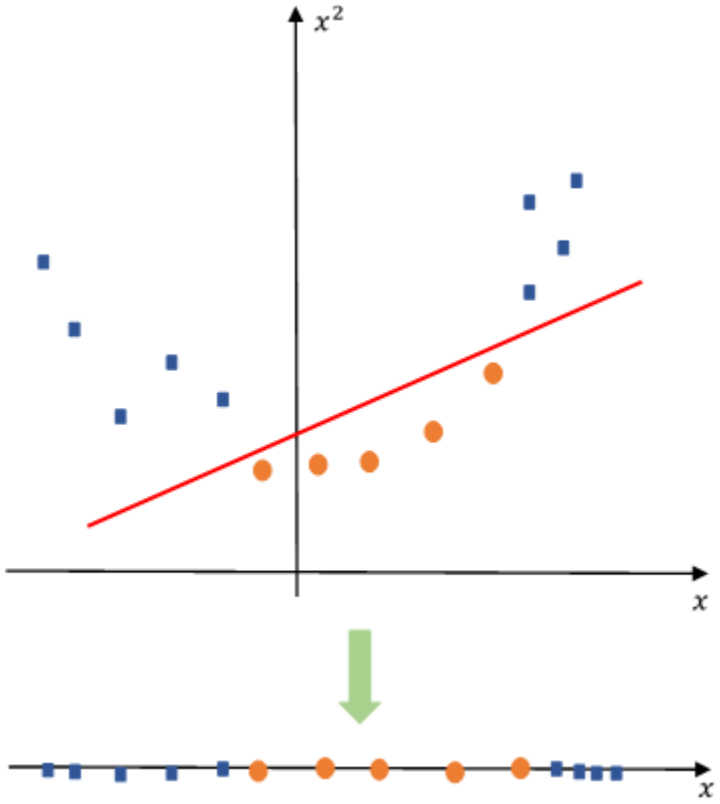
- 2D symmetric matrix.
- Estimating the coherence between all pairs of acquisitions.
- High computing cost.
- Has not been thoroughly investigated.

Theoretical background - Support Vector Machine



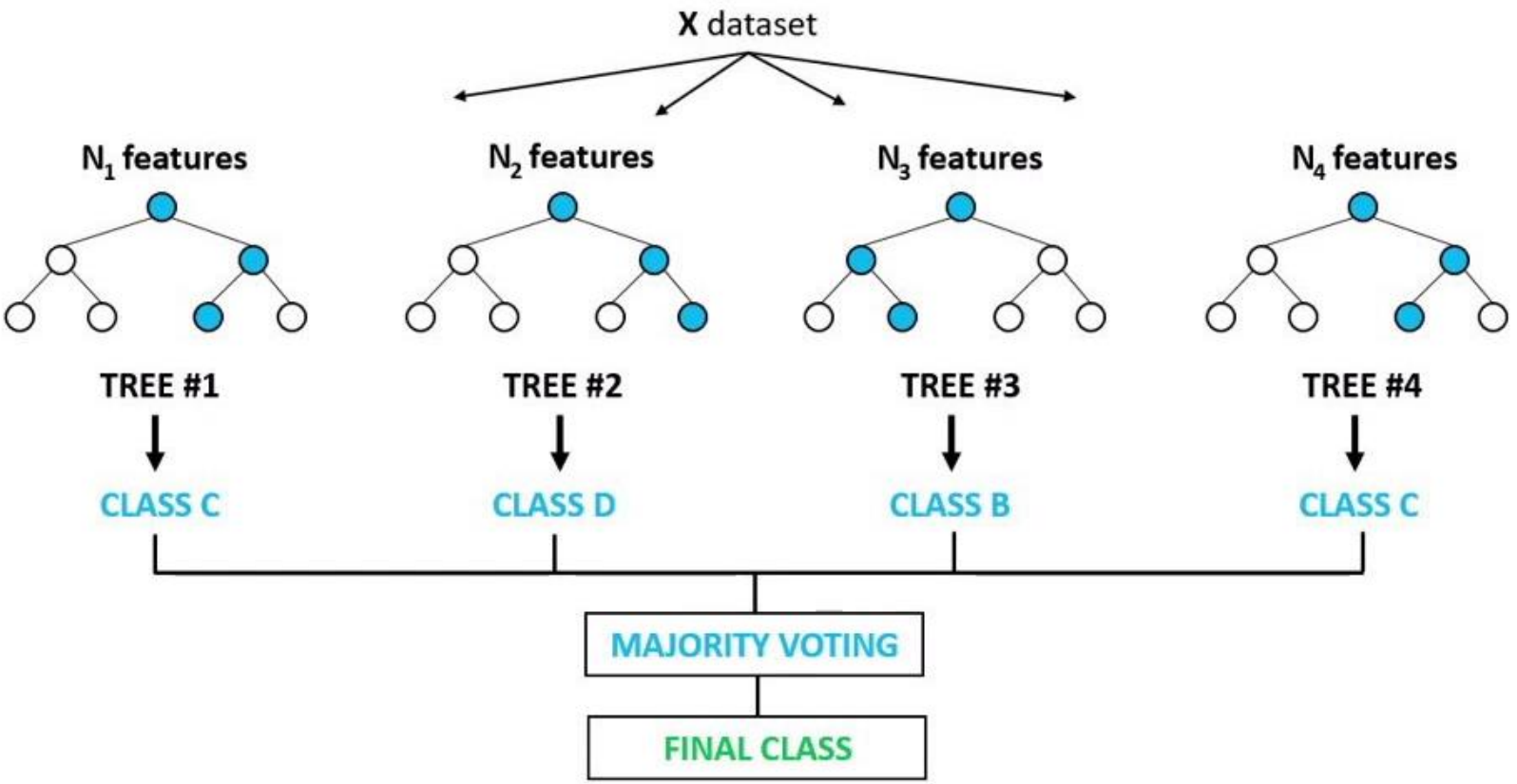
Among the set of hyperplanes, choose the one that maximizes the margin

Theoretical background - Support Vector Machine (The Nonlinear Case)



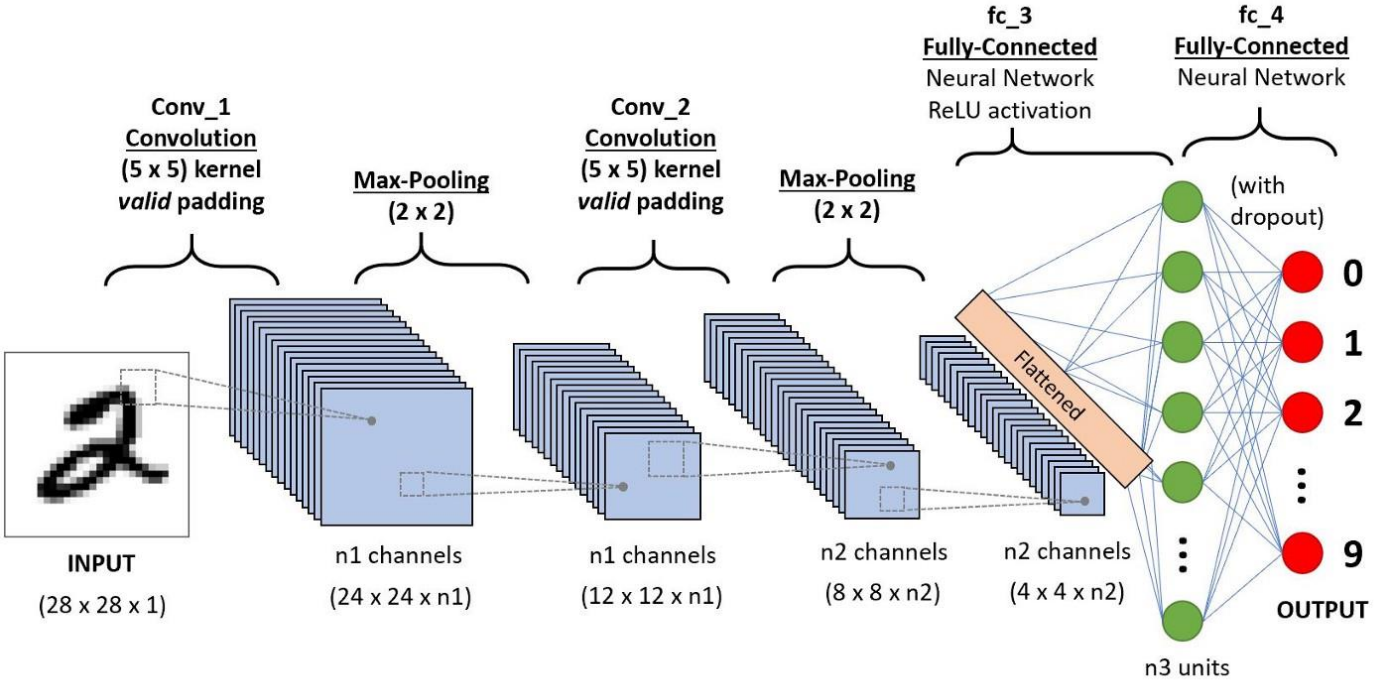
A Kernel Trick is a simple method where a Non-Linear data is projected onto a higher dimension space to make it easier to classify the data where it could be linearly divided by a plane.

Theoretical background - Random Forest



The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

Theoretical background - Convolutional Neural Network



A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers.

Methodology -
Creating data table from the Theia snow collection

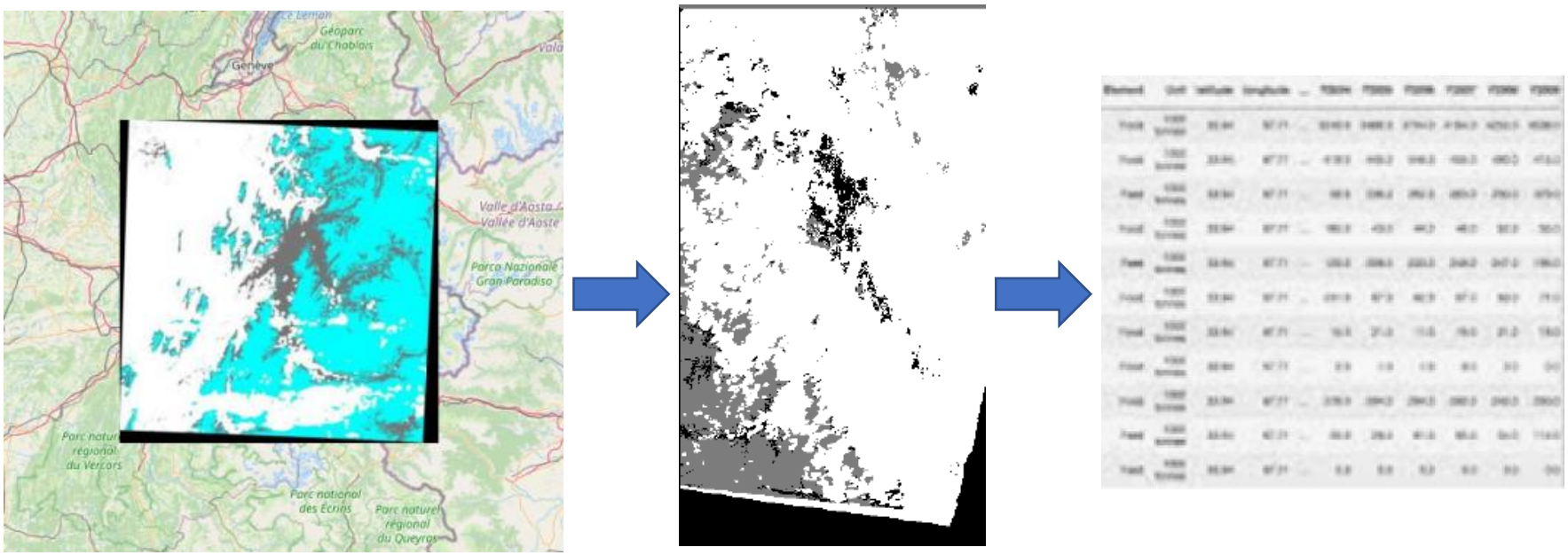


Fig. 9. Data table creation pipeline.

- Optical satellite data is sparse.
- The process of converting a description of a location to a location on the earth's surface is known as georeferencing.
- NSBAS is the satellite radar data processing tool for the analysis of ground displacements.

Methodology - Data masking

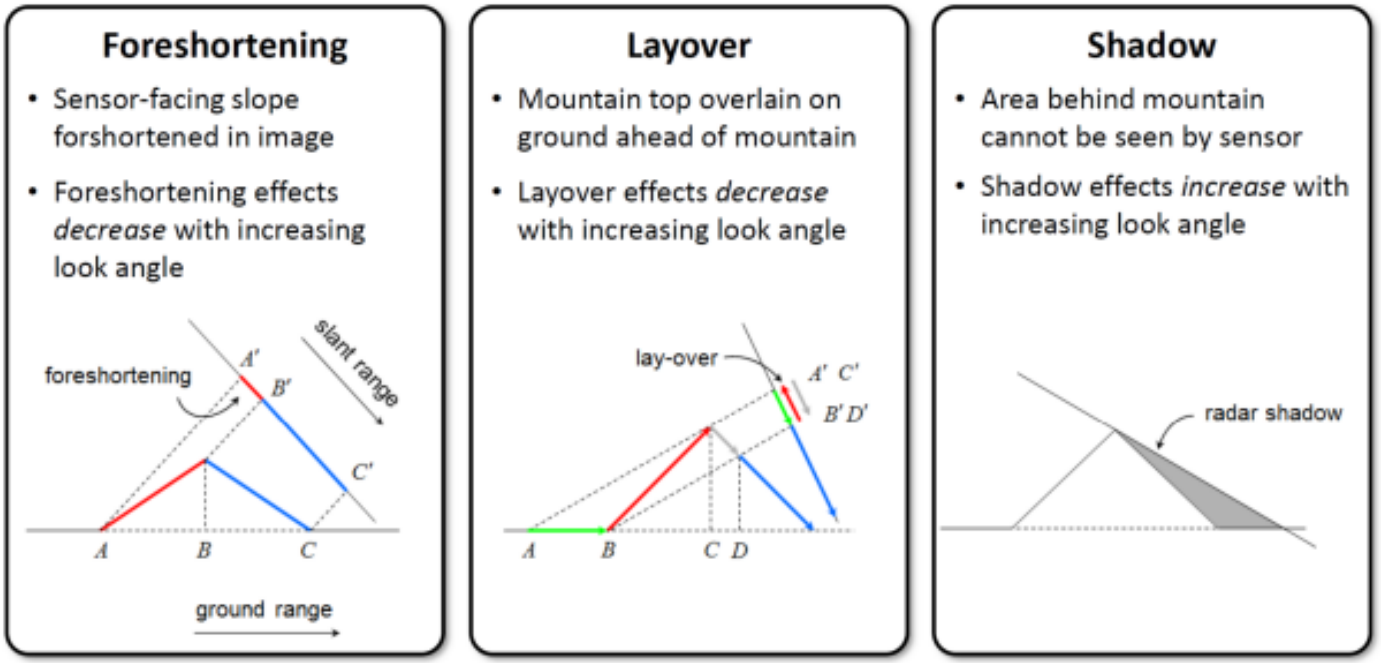


Fig. 10. Distortions induced by side-looking SAR. Ground points a, b, c are 'seen' by radar as points a', b', c' in the slant range. (Figure from Franz J. Meyer)

- SAR acquisitions over complex terrain are often affected by geometric distortions.
- Data masking is required.

Methodology - Computing Coherence Matrices

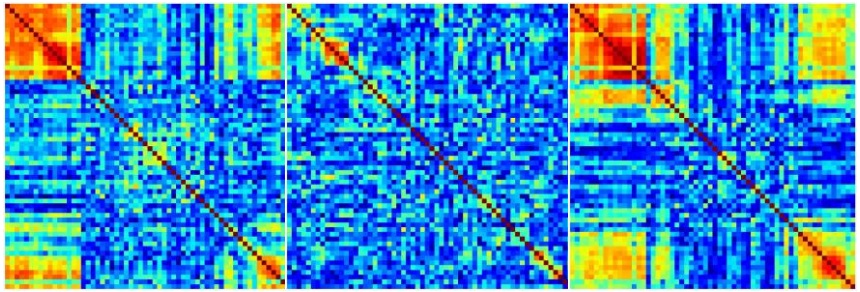


Fig. 11. InSAR coherence matrices with the corresponding ‘no snow’ status

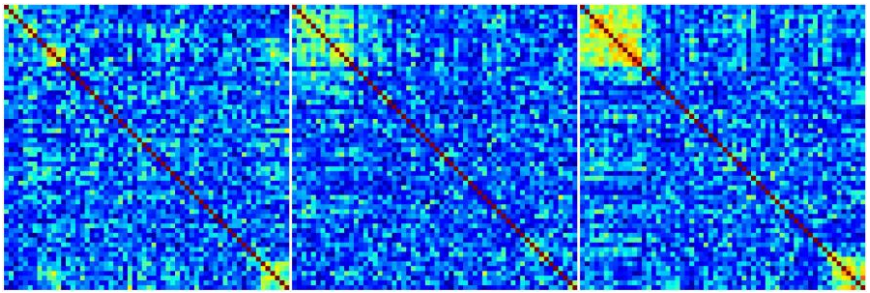


Fig. 12. InSAR coherence matrices with the corresponding ‘snow’ status

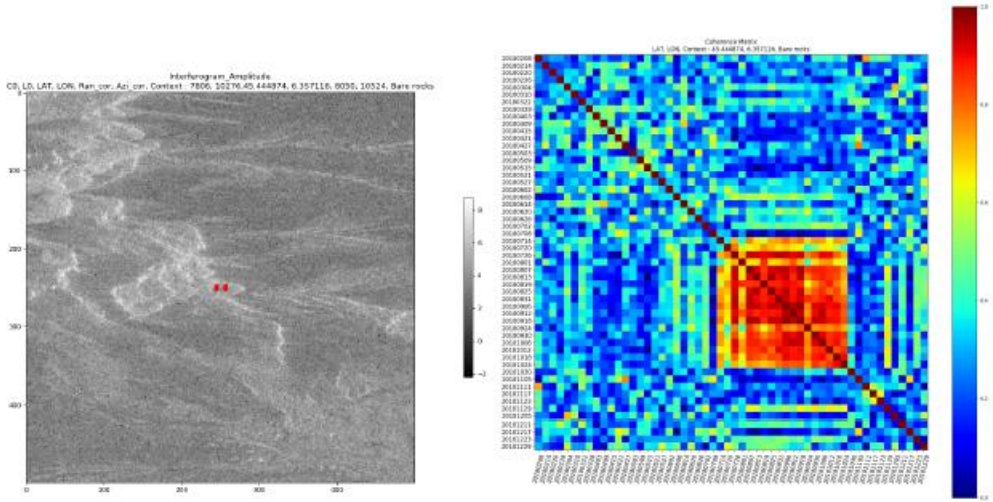


Fig. 13. Example of InSAR coherence matrices with the corresponding average SAR amplitude image of the study area.

Methodology - Binary classification

- Dataset
  - Computed InSAR coherence matrix images
    - 1000 ‘snow’
    - 1000 ‘no snow’
- Library
  - Keras
  - Sklearn
  - Skimage
  - Cv2
- Process
  - Taking input
  - Model construction
  - Model training
  - Model testing

Methodology - Support Vector Machine

Taking input

```
Categories=['nosnow','snow']

flat_data_arr=[] # input array
target_arr=[] # output array

# path which contains all the categories of images
datadir=f'/home/smithtape/Desktop/internship/internship_2021/coding/task13_MLs/5_svm'

for i in Categories:

    print(f'loading... category : {i}')
    path=os.path.join(datadir,i)
    for img in os.listdir(path):
        img_array=imread(os.path.join(path,img))
        img_resized=resize(img_array,(224,224,3))
        flat_data_arr.append(img_resized.flatten())
        target_arr.append(Categories.index(i))
    print(f'loaded category:{i} successfully')
flat_data=np.array(flat_data_arr)
target=np.array(target_arr)
df=pd.DataFrame(flat_data) #dataframe
df['Target']=target
x=df.iloc[:,:-1] #input data
y=df.iloc[:,-1] #output data
```

Methodology - Support Vector Machine

Model construction

```
from sklearn import svm
svc=svm.SVC(kernel='rbf',probability=True)
```

Model training

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=77,stratify=y)
print('Splitted Successfully')
svc.fit(x_train,y_train)
print('The Model is trained well with the given images')
```

Model testing

```
y_pred=svc.predict(x_test)
print("The predicted Data is :")
print(y_pred)
print("The actual data is:")
print(np.array(y_test))
print(f"The model is {accuracy_score(y_pred,y_test)*100}% accurate")
```

Methodology - Random Forest

Taking input

```
Categories=['nosnow','snow']

flat_data_arr=[] # input array
target_arr=[] # output array

# path which contains all the categories of images
datadir=f'/home/smithtape/Desktop/internship/internship_2021/coding/task13_MLs/5_svm'

for i in Categories:

    print(f'loading... category : {i}')
    path=os.path.join(datadir,i)
    for img in os.listdir(path):
        img_array=imread(os.path.join(path,img))
        img_resized=resize(img_array,(224,224,3))
        flat_data_arr.append(img_resized.flatten())
        target_arr.append(Categories.index(i))
    print(f'loaded category:{i} successfully')
flat_data=np.array(flat_data_arr)
target=np.array(target_arr)
df=pd.DataFrame(flat_data) #dataframe
df['Target']=target
x=df.iloc[:,:-1] #input data
y=df.iloc[:,-1] #output data
```

Methodology - Random Forest

Model construction

```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=100)
```

Model training

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=77,stratify=y)
print('Splitted Successfully')
clf.fit(x_train,y_train)
print('The Model is trained well with the given images')
```

Model testing

```
from sklearn.metrics import accuracy_score

y_pred=clf.predict(x_test)
print("The predicted Data is :")
print(y_pred)
print("The actual data is:")
print(np.array(y_test))
print(f"The model is {accuracy_score(y_pred,y_test)*100}% accurate")
```

Methodology - Convolutional Neural Network

Taking input

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# create data generator
train_augmentation = ImageDataGenerator(featurewise_center=True, #featurewise_center transfo
                                        validation_split=0.2)
# specify imagenet mean values for centering
# [123.68, 116.779, 103.939] is from ImageNet, we don not need to change it when we train ou
train_augmentation.mean = [123.68, 116.779, 103.939]
# prepare iterator
train_generator = train_augmentation.flow_from_directory(
  "/home/smithtape/Desktop/internship/internship_2021/coding/task13_MLs/4_cnn/dataset",
  class_mode='binary',
  subset = "training",
  target_size=(img_height, img_width),
  batch_size=batch_size)

validation_generator = train_augmentation.flow_from_directory(
  "/home/smithtape/Desktop/internship/internship_2021/coding/task13_MLs/4_cnn/dataset",
  class_mode='binary',
  subset="validation",
  target_size=(img_height, img_width),
  batch_size=batch_size)
```

Methodology - Convolutional Neural Network

Model construction

```
from keras.applications.vgg16 import VGG16
from keras.optimizers import SGD
from keras.layers import Flatten
from keras.layers import Dense
from keras.models import Model

# include_top=False beacuse I leave out the last fully connected layer
model = VGG16(include_top=False, input_shape=(img_height, img_width, 3))

# since we don't have to train all the layers, we make them non_trainable
for layer in model.layers:
 layer.trainable = False

# flatten the output layer to 1 dimension
flat1 = Flatten()(model.layers[-1].output)
# add a fully connected layer with 128 hidden units and ReLU activation
class1 = Dense(128, activation='relu')(flat1)
# add a final sigmoid layer for classification
output = Dense(1, activation='sigmoid')(class1)
# define new model
model = Model(inputs=model.inputs, outputs=output)
# compile model
opt = SGD(learning_rate=0.001, momentum=0.9)

model.compile(optimizer=opt,
              loss = 'binary_crossentropy',
              metrics=['accuracy'])
```

Methodology - Convolutional Neural Network

Model training

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# create data generator
train_augmentation = ImageDataGenerator(featurewise_center=True, #feat
                                        validation_split=0.2)
```

```
history = model.fit(train_generator,
                    steps_per_epoch = len(train_generator),
                    validation_data = validation_generator,
                    validation_steps = len(validation_generator),
                    epochs=18,
                    verbose=1)
```

Model testing

```
_, acc = model.evaluate(validation_generator, steps = len(validation_generator), verbose=0)
print('accuracy > %.3f%%' % (acc * 100.0))
```

Results and Discussion - Models Performance

Accuracy : The fraction of all predictions that are correct. It can be computed for a class or all classes.

$$Acc_{All} \triangleq \frac{\sum_{i=1}^{L} C_{ii}}{\sum_{i=1}^{L} G_i}$$

Where C_{ij} is the number of pixels having a ground truth label i and being predicted as j, G_i is the total number of pixels labeled with i and L is the number of classes

Results and Discussion - Results

	Validation set accuracy
CNN (VGG16)	*62.3% ± 1.0%*
CNN (VGG19)	*59.8% ± 1.0%*
SVM (with Linear Kernel)	*64.5% ± 1.0%*
SVM (with Radial Basis Function Kernel)	*74.5% ± 1.0%*
SVM (with Polynomial Kernel)	*73.5% ± 1.0%*
Random forest	*71.3% ± 1.0%*

Table. 1. Overall comparison of models' performances.

Results and Discussion - Discussion

- Nonlinear Support Vector Machine is the better choice of algorithm to achieve accuracy in binary image classification of InSAR coherence matrix image.
- The performance of Random forest is also very interesting, it's between Kernelized SVM and Linear SVM.
- CNN performs poorly compared to other machine learning models.
- I reduce the resolution of input images; it can mean loss of information.
- From the experiments, CNN, SVM and Random forest' performances are all above 50%. Using coherence matrix computed from InSAR coherence as an image-like input data, It means full temporal SAR coherence matrix representation has potential in the field of snow detection using satellite imagery

Conclusion

- This method recently created was successful in matching snow status to multi-temporal coherence matrices.
- The landscape's temporal variations carry crucial information about the terrain type being studied.
- Using multi-temporal coherence matrices to apply machine learning and deep learning models to interferometric SAR data, for snow detection prove effective.
- The best performing models being the Non-Linear SVM.