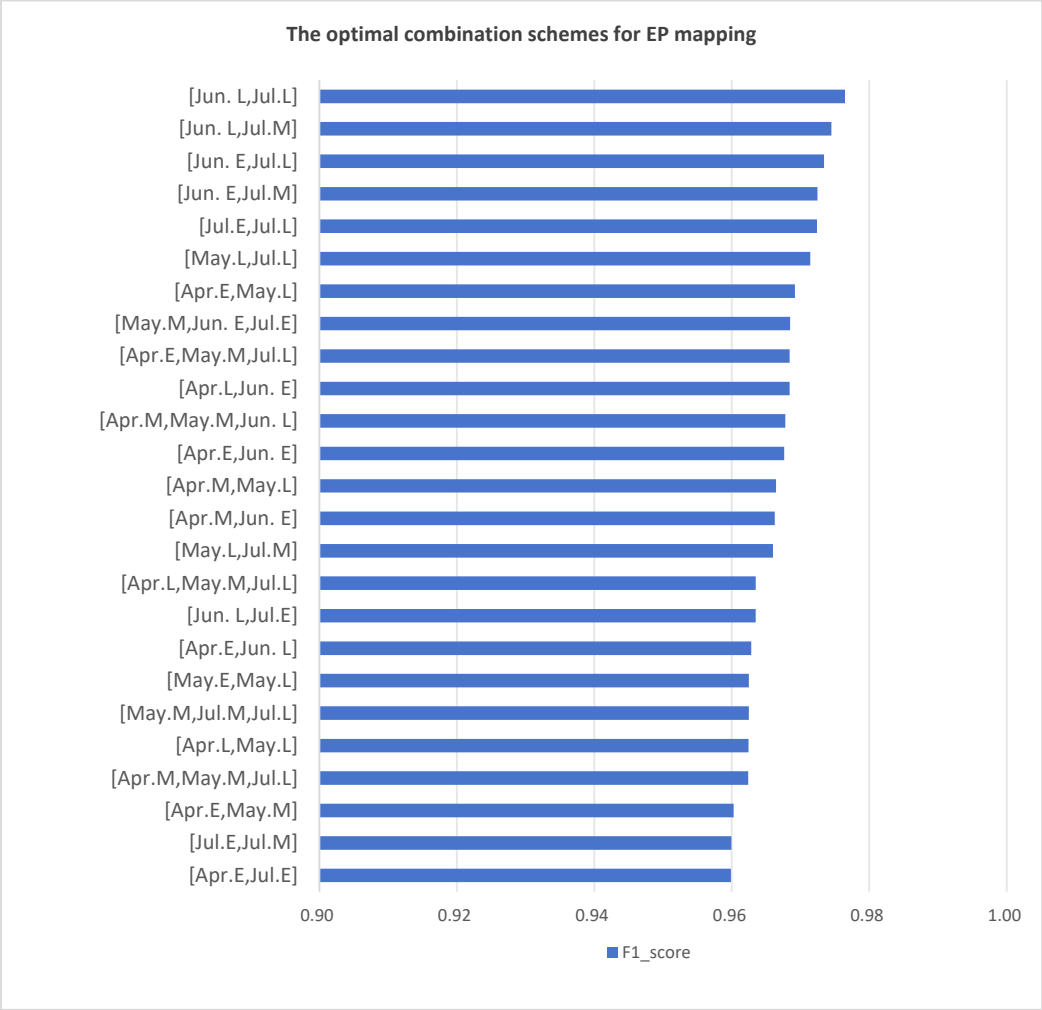


Figure S1. The optimal combination schemes for early cropping paddy (EP) mapping with the F1_score aim of 0.96

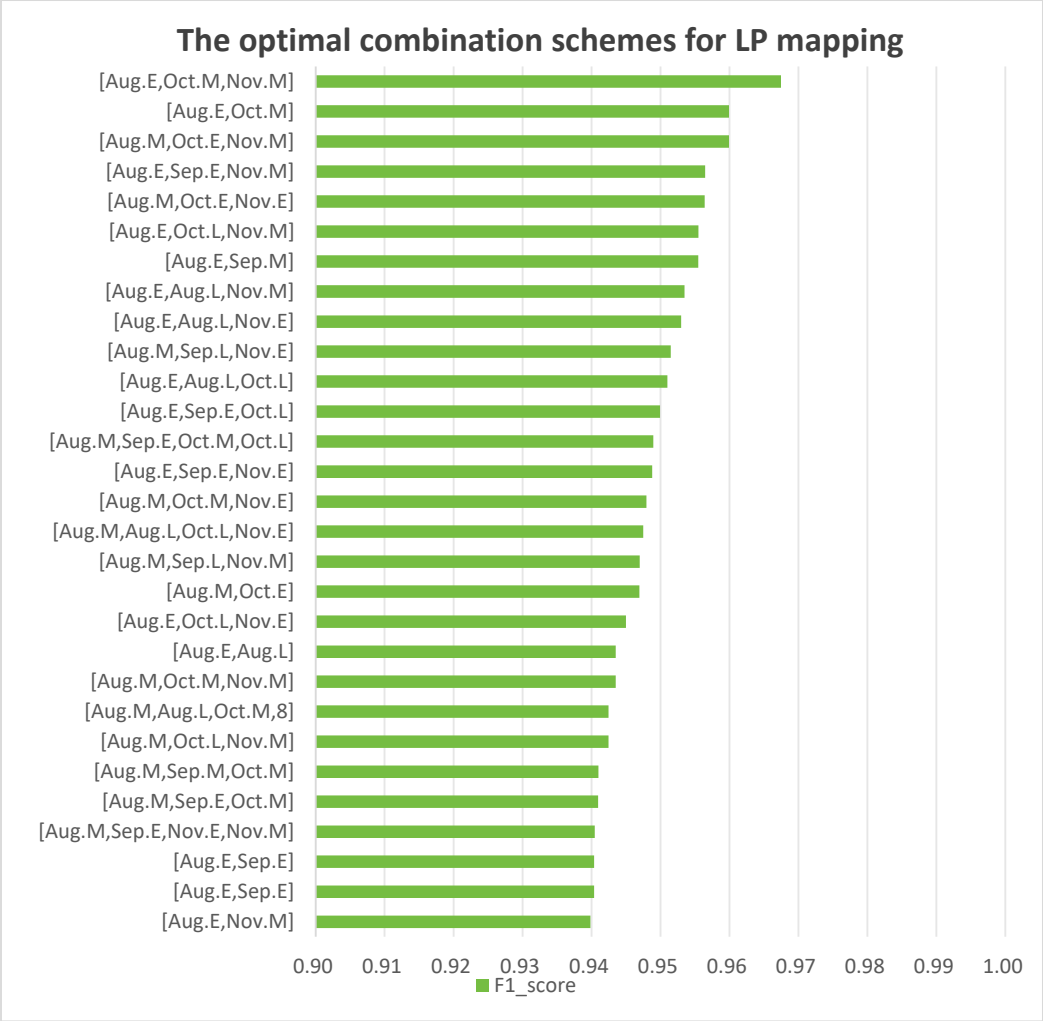


Figure S2. The optimal combination schemes for late cropping paddy (LP) mapping with the F1_score aim of 0.94

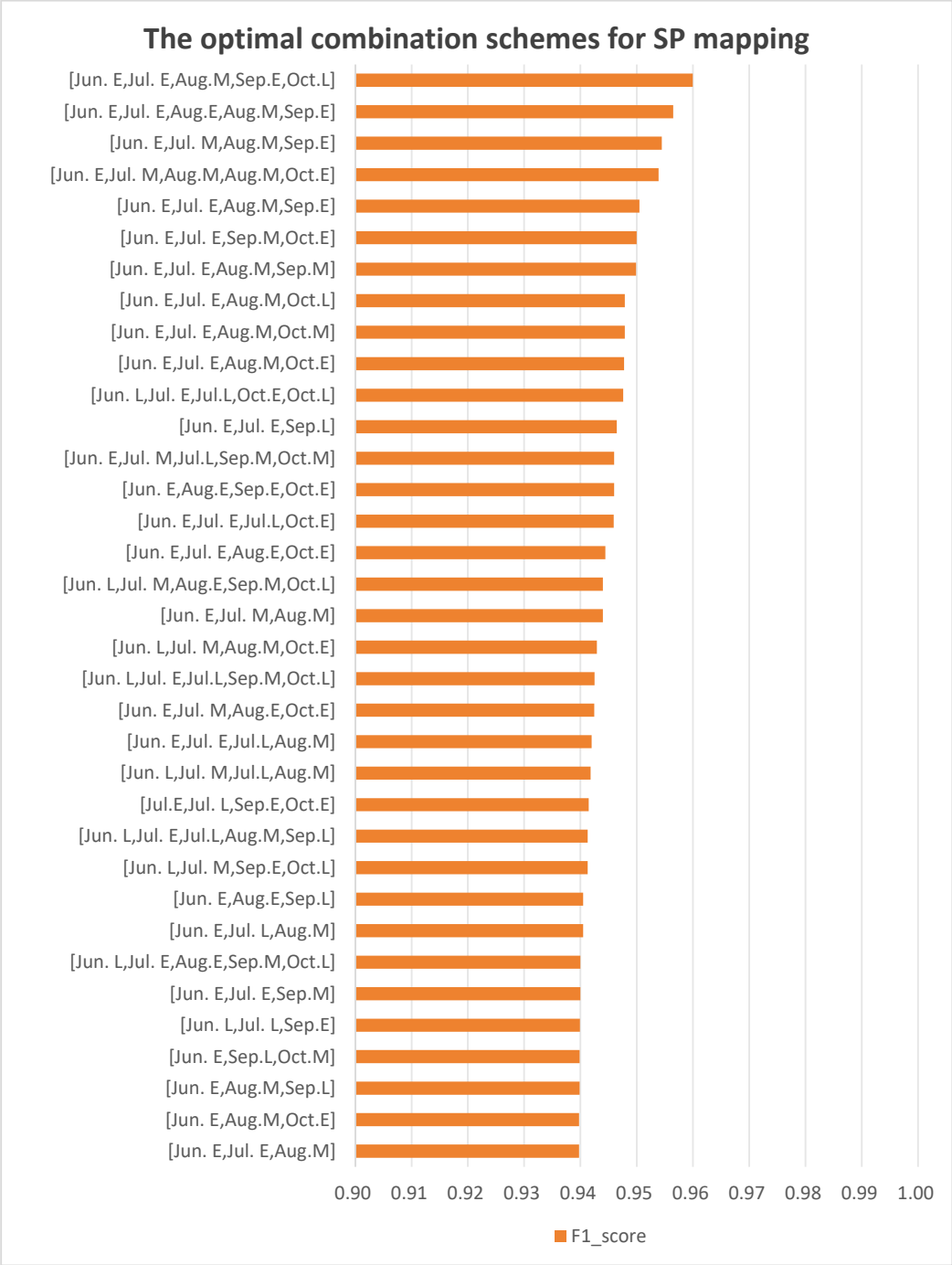


Figure S3. The optimal combination schemes for single cropping paddy (SP) mapping with the F1_score aim of 0.94.

Table S1. GEE CODE for processing Sentinel-2 time series images

```
var startDate='***'
var endDate='***'
var foldername='***'
var bounds='***'
var selectedbands_index=['NDVI','NDWI','LSWI','NDRE','EVI','SAVI','BSI','GCVI','PSRI','SIPI']
var selectedbands=['B2','B3','B4','B5','B6','B7','B8','B8A','B11','B12']

//**1.select image from tile;
//**2.cloud mask and cloud shade mask from S2_CLOUD_PROBABILITY;
//**3.calculation of the index
function cal_index(image) {
   var ndvi=image.normalizedDifference(['B8','B4']).multiply(1000).toInt16().rename('NDVI');
   var ndwi=image.normalizedDifference(['B3','B8']).multiply(1000).toInt16().rename('NDWI');
   var lswi=image.normalizedDifference(['B8','B11']).multiply(1000).toInt16().rename('LSWI');
   var ndre=image.normalizedDifference(['B6','B5']).multiply(1000).toInt16().rename('NDRE');
   var evi = image.expression('2.5 * ((NIR-RED) / (NIR + 6*RED - 7.5*BLUE + 1))',
                                          {'NIR': image.select('B8'),
                                            'RED': image.select('B4'),
                                            'BLUE': image.select('B2')}).multiply(1000).toInt16().rename('EVI');
   var savi=image.expression('1.5 * ((NIR-RED) / (NIR + RED +0.5))',
                                          {'NIR': image.select('B8'),
                                            'RED': image.select('B4')}).multiply(1000).toInt16().rename('SAVI');
   var bsi=image.expression('((SWIR+RED)-(NIR+BLUE)) / ((SWIR+RED)+(NIR+BLUE))',
                                          {'SWIR':image.select('B11'),
                                            'NIR': image.select('B8'),
                                            'RED': image.select('B4'),
                                            'BLUE':image.select('B2')}).multiply(1000).toInt16().rename('BSI');
   var gcvi=image.expression('(NIR/GREEN)-1 ',
                                          {'NIR': image.select('B8'),
                                            'GREEN': image.select('B3')}).multiply(1000).toInt16().rename('GCVI');
   var psri=image.expression('(RED-BLUE)/NIR',
                                          {'NIR': image.select('B8'),
                                            'RED': image.select('B4'),
                                            'BLUE': image.select('B2')}).multiply(1000).toInt16().rename('PSRI');
   var sipi=image.expression('(NIR-BLUE)/(NIR+RED)',
                                          {'NIR': image.select('B8'),
                                            'RED': image.select('B4'),
                                            'BLUE': image.select('B2')}).multiply(1000).toInt16().rename('SIPI');
   return image.addBands(ndvi).addBands(ndwi).addBands(lswi).addBands(ndre).addBands(evi).
               addBands(savi).addBands(bsi).addBands(gcvi).addBands(psri).addBands(sipi)
}

//mask cloud and cloudshade,the value of 65 is from the sample code
function maskImage(image) {
   var prob = image.select('probability');
   var isCloud = prob.gt(65);
   isCloud = isCloud.focalMin(3).focalMax(16);
   isCloud = isCloud.reproject({crs: prob.projection(), scale: 10});
   var shadowAzimuth = ee.Number(90)
         .subtract(ee.Number(image.get('MEAN_SOLAR_AZIMUTH_ANGLE')));
   isCloud = isCloud.directionalDistanceTransform(shadowAzimuth, 50);
   isCloud = isCloud.select('distance').mask().rename('CloudMask');
   return isCloud.not();
}

function indexJoin(collectionA, collectionB, propertyName) {
   var joined = ee.ImageCollection(ee.Join.saveFirst(propertyName).apply({
      primary: collectionA,
      secondary: collectionB,
      condition: ee.Filter.equals({
         leftField: 'system:index',
         rightField: 'system:index'})
   }));
   // Merge the bands of the joined image.
   return joined.map(function(image) {
      return image.addBands(ee.Image(image.get(propertyName)));
   });
}

var s2_dataset = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
                              .filterDate(startDate, endDate).filterBounds(bounds)
                              //.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',50))
                                                                  //.filter(ee.Filter.eq('MGRS_TILE', tile))
var s2c_dataset= ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY')
                              .filterDate(startDate, endDate)
                              .filterBounds(bounds);
var withCloudProbability = indexJoin(s2_dataset, s2c_dataset, 'cloud_probability');
var masked = ee.ImageCollection(withCloudProbability.map(maskImage));
var s2_dataset_WithMask = indexJoin(s2_dataset,masked,'cloud_mask');
var s2_dataset_CloudMasked = s2_dataset_WithMask.map(function(img){
      var mask = img.select('CloudMask')
      return img.updateMask(mask)
                  .divide(10000)
```

```
                    .select('B.*')
                    .toFloat()
                    .copyProperties(img, ["system:time_start"])
})

var index_dataset=ee.ImageCollection(s2_dataset_CloudMasked.map(cal_index))
                            .select(selectedbands_index)
print(index_dataset)

//export the result
var listofimage=index_dataset.aggregate_array('system:index')
listofimage.evaluate(function (indexes) {
        indexes.forEach(function (index) {
            var image = index_dataset
                .filterMetadata('system:index', 'equals', index)
                .first();
            var description = 'S2_index_' + index.replace(/[^\w]/, '_');
            Export.image.toDrive({
                image: image,
                description: description,
                folder: foldername,
                scale: 10,
                maxPixels: 3784216672400,
                region:bounds
            });
        })
})
```