Table of Contents

Software Guide for Tau Reconstruction UPDATED

The goal of this page is to document the usage and creation of hadronic tau-jet candidates from ParticleFlow jets.

Contact : **Isobel Ojalvo**, **Michal Bluj**, **Yuta Takahashi**, **Arun Nayak**
hn-cms-tauid@cernNOSPAMPLEASE.ch

Tau Reconstruction

The minimal recommended tags for tau reconstruction can be found below.

If you want to use on top, the analysis tools (PAT) please check the recommended tags here SWGuidePATRecipes.

A dedicated analysis package for Ztautau, Wtaunu and HiggsTauTau studies can be found here SWGuideTauAnalysis.

The recommended working points by the Tau POG can be found in TauIDRecommendation, TauIDRecommendation13TeV.

UPDATED The github area for tau code development is https://github.com/cms-tau-pog

An introduction to how to use git for development is in this talk

Check the section on discriminators below to understand their proper usage.

Rerunning of the tau ID on AOD event content

The tau collections contained in the simulated samples are often outdated. The most recent tau ID is in general contained in the releases, so to benefit from all new features and bugfixes, you should re-run the PFTau sequence on RECO/AOD. Because the the most up-to-date software is almost always contained in the production releases, **there is no need to merge in any code from other repositories**.

To re-run the tau sequence in, you need to add following few lines to your config file

```
process.load("RecoTauTag.Configuration.RecoPFTauTag_cff") #loading the configuration

from PhysicsTools.PatAlgos.tools.tauTools import * # to get the most up-to-date discriminators wh
switchToPFTauHPS(process) # this line and the one above can be ignored when running on reco::PFTa
...
process.p = cms.Path(   #adding to the path
....
        process.PFTau*
....
)
```

When running in *un-scheduled mode* it is enough to add `process.load("RecoTauTag.Configuration.RecoPFTauTag_cff")` to the config file.

Rerunning of the tau ID on MiniAOD event content in CMSSW 8_0_X NEW

While it is not possible to fully rebuild taus from jets given MiniAOD event content, it is possible to recompute the BDT output of both isolation and anti-electron discriminators for new trainings made available

starting from CMSSW 8_0_X onwards. While for releases from 8_1_X onwards the necessary infrastructure is included in official CMSSW releases, for 8_0_X one has to merge developments from the following cms-tau-pog branch: `cms-tau-pog/CMSSW_8_0_X_tau-pog_tauIDOnMiniAOD-legacy-backport-81X` . In your CMSSW_8_0_X/src/ area do:

```
git cms-merge-topic -u cms-tau-pog:CMSSW_8_0_X_tau-pog_tauIDOnMiniAOD-legacy-backport-81X     # c
git cms-merge-topic -u cms-tau-pog:CMSSW_8_0_X_tau-pog_tauIDOnMiniAOD-legacy-backport-81Xv2   # c
```

Note that the backport branch of choice depends on the CMSSW version that you are using. The `-u` is necessary to avoid that git checks out all packages that depend on any of the ones touched in this branch since this would lead to a very long compilation. Then compile everything.

In order to be able to access the latest and greatest BDT output for the isolation discriminators and save it in your ntuples, you need to add a sequence to your python config file and some code to your analyzer. You can find an example analyzer and the corresponding config file in RecoTauTag/RecoTau/test. Note that the backport include two features to facilitate the acces to the new tau-ID discriminators:

- From the example you can see, how to embed the new discriminators into a re-created *pat::Tau* collection. FROm that point on you can access the new discriminators as usual from their names as parts of the new pat::Tau object.
- In case you want to re-run the the anti-electron MVA6 discriminator from miniAOD event content, you might face the problem, that one low level varibale, the phi angle at the ECAL entrance is not saved in the miniAOD. In this case you can bypass this flaw by recalculating it on-the-fly, by track extrapolation. For this purpose in the configuration of the discriminator set the parameter *usePhiAtEcalEntranceExtrapolation* to True.

Below, a code example for including the new training with old decay modes w/ embedding into a new *pat::Tau* collection is shown. The procedure is the same for the training with new decay modes. Please refer to the TauIDRecommendation13TeV TWiki for the lines that need to be changed.

Additions to python config file for new MVA isolation ▸ Hide ▾

```
from RecoTauTag.RecoTau.TauDiscriminatorTools import noPrediscriminants
process.load('RecoTauTag.Configuration.loadRecoTauTagMVAsFromPrepDB_cfi')
from RecoTauTag.RecoTau.PATTauDiscriminationByMVAIsolationRun2_cff import *

process.rerunDiscriminationByIsolationMVArun2v1raw = patDiscriminationByIsolationMVArun2v1raw.clo
   PATTauProducer = cms.InputTag('slimmedTaus'),
   Prediscriminants = noPrediscriminants,
   loadMVAfromDB = cms.bool(True),
   mvaName = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMwLT2016v1"), # name of the training you wan
   mvaOpt = cms.string("DBoldDMwLT"), # option you want to use for your training (i.e., which var
   requireDecayMode = cms.bool(True),
   verbosity = cms.int32(0)
)

process.rerunDiscriminationByIsolationMVArun2v1VLoose = patDiscriminationByIsolationMVArun2v1VLoo
   PATTauProducer = cms.InputTag('slimmedTaus'),
   Prediscriminants = noPrediscriminants,
   toMultiplex = cms.InputTag('rerunDiscriminationByIsolationMVArun2v1raw'),
   key = cms.InputTag('rerunDiscriminationByIsolationMVArun2v1raw:category'),
   loadMVAfromDB = cms.bool(True),
   mvaOutput_normalization = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMwLT2016v1_mvaOutput_normali
   mapping = cms.VPSet(
      cms.PSet(
         category = cms.uint32(0),
         cut = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMwLT2016v1_WPEff90"), # this is the name o
         variable = cms.string("pt"),
      )
   )
```

```
)

# here we produce all the other working points for the training
process.rerunDiscriminationByIsolationMVArun2v1Loose = process.rerunDiscriminationByIsolationMVAr
process.rerunDiscriminationByIsolationMVArun2v1Loose.mapping[0].cut = cms.string("RecoTauTag_tauI
process.rerunDiscriminationByIsolationMVArun2v1Medium = process.rerunDiscriminationByIsolationMVA
process.rerunDiscriminationByIsolationMVArun2v1Medium.mapping[0].cut = cms.string("RecoTauTag_tau
process.rerunDiscriminationByIsolationMVArun2v1Tight = process.rerunDiscriminationByIsolationMVAr
process.rerunDiscriminationByIsolationMVArun2v1Tight.mapping[0].cut = cms.string("RecoTauTag_tauI
process.rerunDiscriminationByIsolationMVArun2v1VTight = process.rerunDiscriminationByIsolationMVA
process.rerunDiscriminationByIsolationMVArun2v1VTight.mapping[0].cut = cms.string("RecoTauTag_tau
process.rerunDiscriminationByIsolationMVArun2v1VVTight = process.rerunDiscriminationByIsolationMV
process.rerunDiscriminationByIsolationMVArun2v1VVTight.mapping[0].cut = cms.string("RecoTauTag_ta

# this sequence has to be included in your cms.Path() before your analyzer which accesses the new
process.rerunMvaIsolation2SeqRun2 = cms.Sequence(
   process.rerunDiscriminationByIsolationMVArun2v1raw
   *process.rerunDiscriminationByIsolationMVArun2v1VLoose
   *process.rerunDiscriminationByIsolationMVArun2v1Loose
   *process.rerunDiscriminationByIsolationMVArun2v1Medium
   *process.rerunDiscriminationByIsolationMVArun2v1Tight
   *process.rerunDiscriminationByIsolationMVArun2v1VTight
   *process.rerunDiscriminationByIsolationMVArun2v1VVTight
)


# embed new id's into new tau collection
embedID = cms.EDProducer("PATTauIDEmbedder",
   src = cms.InputTag('slimmedTaus'),
   tauIDSources = cms.PSet(
      byIsolationMVArun2v1DBoldDMwLTrawNew = cms.InputTag('rerunDiscriminationByIsolationMVArun2v
      byVLooseIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAru
      byLooseIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVArun
      byMediumIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAru
      byTightIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVArun
      byVTightIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAru
      byVVTightIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAr
      . . . (other discriminators like anti-electron),
      ),
   )
setattr(process, "NewTauIDsEmbedded", embedID)

process.p = cms.Path(
   . . . (other processes)
   *process.rerunMvaIsolation2SeqRun2
   *getattr(process, "NewTauIDsEmbedded")
   . . . (for example you ntuple creation process)
)
```

The python configuration to be added to include new trainings of the anti-electron discriminator is shown below.

Additions to python config file for new anti-electron MVA ▸ Hide ▾

```
process.load('RecoTauTag.Configuration.loadRecoTauTagMVAsFromPrepDB_cfi')
from RecoTauTag.RecoTau.PATTauDiscriminationAgainstElectronMVA6_cfi import *

process.rerunDiscriminationAgainstElectronMVA6 = patTauDiscriminationAgainstElectronMVA6.clone(
   PATTauProducer = cms.InputTag('slimmedTaus'),
   Prediscriminants = noPrediscriminants,
   #Prediscriminants = requireLeadTrack,
   loadMVAfromDB = cms.bool(True),
   returnMVA = cms.bool(True),
   method = cms.string("BDTG"),
   mvaName_NoEleMatch_woGwoGSF_BL = cms.string("RecoTauTag_antiElectronMVA6v1_gbr_NoEleMatch_woGw
   mvaName_NoEleMatch_wGwoGSF_BL = cms.string("RecoTauTag_antiElectronMVA6v1_gbr_NoEleMatch_wGwoG
   mvaName_woGwGSF_BL = cms.string("RecoTauTag_antiElectronMVA6v1_gbr_woGwGSF_BL"),
```

```
   mvaName_wGwGSF_BL = cms.string("RecoTauTag_antiElectronMVA6v1_gbr_wGwGSF_BL"),
   mvaName_NoEleMatch_woGwoGSF_EC = cms.string("RecoTauTag_antiElectronMVA6v1_gbr_NoEleMatch_woGw
   mvaName_NoEleMatch_wGwoGSF_EC = cms.string("RecoTauTag_antiElectronMVA6v1_gbr_NoEleMatch_wGwoG
   mvaName_woGwGSF_EC = cms.string("RecoTauTag_antiElectronMVA6v1_gbr_woGwGSF_EC"),
   mvaName_wGwGSF_EC = cms.string("RecoTauTag_antiElectronMVA6v1_gbr_wGwGSF_EC"),
   minMVANoEleMatchWOgWOgsfBL = cms.double(0.0),
   minMVANoEleMatchWgWOgsfBL  = cms.double(0.0),
   minMVAWOgWgsfBL            = cms.double(0.0),
   minMVAWgWgsfBL             = cms.double(0.0),
   minMVANoEleMatchWOgWOgsfEC = cms.double(0.0),
   minMVANoEleMatchWgWOgsfEC  = cms.double(0.0),
   minMVAWOgWgsfEC            = cms.double(0.0),
   minMVAWgWgsfEC             = cms.double(0.0),
   srcElectrons = cms.InputTag('slimmedElectrons'),
        usePhiAtEcalEntranceExtrapolation = cms.bool(True)
)
# embed new id's into new tau collection
embedID = cms.EDProducer("PATTauIDEmbedder",
   src = cms.InputTag('slimmedTaus'),
   tauIDSources = cms.PSet(
      . . . (other new discriminators like isolation),
      againstElectronMVA6RawNew = cms.InputTag('rerunDiscriminationAgainstElectronMVA6')
      ),
   )
setattr(process, "NewTauIDsEmbedded", embedID)

process.p = cms.Path(
   . . . (other processes)
   *process.rerunDiscriminationAgainstElectronMVA6
   *getattr(process, "NewTauIDsEmbedded")
   . . . (for example you ntuple creation process)
)
```

Please be mindful that if you want to include new isolation and anti-electron discriminators at the same time, things like the PATTauIDEmbedder need only be run once. Just reorder/change the example python configuration snippets accordingly.

Once the new discriminators are embedded into the new tau collection (in this example called "NewTauIDsEmbedded") one can simply retrieve them via the `pat::Tau::tauID` function in a loop over the collection like so:

```
float newIsolationMVArawValue = tau->tauID("byIsolationMVArun2v1DBoldDMwLTrawNew");
float newAntiElectronMVArawValue = tau->tauID("againstElectronMVA6RawNew");
```

Rerunning of the tau ID on MiniAOD event content in CMSSW 9_4_X NEW

While it is not possible to fully rebuild taus from jets given MiniAOD event content, it is possible to recompute the BDT output of the isolation discriminators for new trainings done with the 2017 simulations v1.

In order to be able to access the latest and greatest BDT output for the isolation discriminators and save it in your ntuples, you need to add a sequence to your python config file and some code to your analyzer. The recipe is similar to the one for CMSSW 8_0_X, with the exception that the discriminator payload needs to be loaded in the python config file.

Most simple and direct recipe can be found in the attached presentation. The steps can be summarised as following the example with 2017v2 inclusion:

1. Download and place in YOUR python directory module that constructs a configuration for rerunning the tau sequence for different discriminators runTauIdMVA.py script

2. Add to your PSet initialisation

```
from <your path>.runTauIdMVA import *
na = TauIDEmbedder(process, cms, # pass tour process object
    debug=True,
    toKeep = ["2017v2"] # pick the one you need: ["2017v1", "2017v2", "newDM2017v2", "dR0p32017v2
)
na.runTauID()
```

3. Define handles to access discriminators in your analysis module:

```
byIsolationMVArun2017v2DBoldDMwLTraw2017 = cms.string('byIsolationMVArun2017v2DBoldDMwLTraw2017')
byVVLooseIsolationMVArun2017v2DBoldDMwLT2017 = cms.string('byVVLooseIsolationMVArun2017v2DBoldDMw
byVLooseIsolationMVArun2017v2DBoldDMwLT2017 = cms.string('byVLooseIsolationMVArun2017v2DBoldDMwLT
byLooseIsolationMVArun2017v2DBoldDMwLT2017 = cms.string('byLooseIsolationMVArun2017v2DBoldDMwLT20
byMediumIsolationMVArun2017v2DBoldDMwLT2017 = cms.string('byMediumIsolationMVArun2017v2DBoldDMwLT
byTightIsolationMVArun2017v2DBoldDMwLT2017 = cms.string('byTightIsolationMVArun2017v2DBoldDMwLT20
byVTightIsolationMVArun2017v2DBoldDMwLT2017 = cms.string('byVTightIsolationMVArun2017v2DBoldDMwLT
byVVTightIsolationMVArun2017v2DBoldDMwLT2017 = cms.string('byVVTightIsolationMVArun2017v2DBoldDMw
```

4. Add to your sequence the rerunning of tau reconstruction sequence with wanted MVA

```
process.p = cms.Path( <your processes not using new MVAs>
    * process.rerunMvaIsolationSequence
    * process.NewTauIDsEmbedded # *getattr(process, "NewTauIDsEmbedded")
    <rest of your processes>)
```

For those who prefer to stick to the 2016 manner of tau MVA inclusion a code example for including the recent old decay mode 2017v1 and dR=0.3 2017v2 trainings w/ embedding into a new *pat::Tau* collection is shown below. To access the 2017v2 one has to replace "v1" by "v2". Please refer to the TauIDRecommendation13TeV TWiki for the lines that need to be changed.

Additions to python config file for new MVA isolation ▸ Hide ▾

```
from RecoTauTag.RecoTau.TauDiscriminatorTools import noPrediscriminants
process.load('RecoTauTag.Configuration.loadRecoTauTagMVAsFromPrepDB_cfi')
from RecoTauTag.RecoTau.PATTauDiscriminationByMVAIsolationRun2_cff import *

def loadMVA_WPs_run2_2017(process):

      for training, gbrForestName in tauIdDiscrMVA_trainings_run2_2017.items():

         process.loadRecoTauTagMVAsFromPrepDB.toGet.append(
            cms.PSet(
               record = cms.string('GBRWrapperRcd'),
               tag = cms.string("RecoTauTag_%s%s" % (gbrForestName, tauIdDiscrMVA_2017_version)),
               label = cms.untracked.string("RecoTauTag_%s%s" % (gbrForestName, tauIdDiscrMVA_201
            )
         )

         for WP in tauIdDiscrMVA_WPs_run2_2017[training].keys():
            process.loadRecoTauTagMVAsFromPrepDB.toGet.append(
               cms.PSet(
                  record = cms.string('PhysicsTGraphPayloadRcd'),
                  tag = cms.string("RecoTauTag_%s%s_WP%s" % (gbrForestName, tauIdDiscrMVA_2017_ve
                  label = cms.untracked.string("RecoTauTag_%s%s_WP%s" % (gbrForestName, tauIdDisc
               )
            )

         process.loadRecoTauTagMVAsFromPrepDB.toGet.append(
            cms.PSet(
               record = cms.string('PhysicsTFormulaPayloadRcd'),
               tag = cms.string("RecoTauTag_%s%s_mvaOutput_normalization" % (gbrForestName, tauId
```

```
                label = cms.untracked.string("RecoTauTag_%s%s_mvaOutput_normalization" % (gbrFores
            )
)


# 2017 v1
tauIdDiscrMVA_trainings_run2_2017 = {
  'tauIdMVAIsoDBoldDMwLT2017' : "tauIdMVAIsoDBoldDMwLT2017",
  }
tauIdDiscrMVA_WPs_run2_2017 = {
  'tauIdMVAIsoDBoldDMwLT2017' : {
    'Eff95' : "DBoldDMwLTEff95",
    'Eff90' : "DBoldDMwLTEff90",
    'Eff80' : "DBoldDMwLTEff80",
    'Eff70' : "DBoldDMwLTEff70",
    'Eff60' : "DBoldDMwLTEff60",
    'Eff50' : "DBoldDMwLTEff50",
    'Eff40' : "DBoldDMwLTEff40"
    }
  }
tauIdDiscrMVA_2017_version = "v1"

loadMVA_WPs_run2_2017(process)

process.rerunDiscriminationByIsolationMVArun2v1raw = patDiscriminationByIsolationMVArun2v1raw.clo
   PATTauProducer = cms.InputTag('slimmedTaus'),
   Prediscriminants = noPrediscriminants,
   loadMVAfromDB = cms.bool(True),
   mvaName = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMwLT2017v1"),  # name of the training you wa
   mvaOpt = cms.string("DBoldDMwLTwGJ"), # option you want to use for your training (i.e., which
   requireDecayMode = cms.bool(True),
   verbosity = cms.int32(0)
)

process.rerunDiscriminationByIsolationMVArun2v1VLoose = patDiscriminationByIsolationMVArun2v1VLoo
   PATTauProducer = cms.InputTag('slimmedTaus'),
   Prediscriminants = noPrediscriminants,
   toMultiplex = cms.InputTag('rerunDiscriminationByIsolationMVArun2v1raw'),
   key = cms.InputTag('rerunDiscriminationByIsolationMVArun2v1raw:category'),
   loadMVAfromDB = cms.bool(True),
   mvaOutput_normalization = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMwLT2017v1_mvaOutput_normali
   mapping = cms.VPSet(
      cms.PSet(
         category = cms.uint32(0),
         cut = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMwLT2017v1_WPEff90"), # this is the name o
         variable = cms.string("pt"),
      )
   )
)

# here we produce all the other working points for the training
process.rerunDiscriminationByIsolationMVArun2v1VVLoose = process.rerunDiscriminationByIsolationMV
process.rerunDiscriminationByIsolationMVArun2v1VVLoose.mapping[0].cut = cms.string("RecoTauTag_ta
process.rerunDiscriminationByIsolationMVArun2v1Loose = process.rerunDiscriminationByIsolationMVAr
process.rerunDiscriminationByIsolationMVArun2v1Loose.mapping[0].cut = cms.string("RecoTauTag_tauI
process.rerunDiscriminationByIsolationMVArun2v1Medium = process.rerunDiscriminationByIsolationMVA
process.rerunDiscriminationByIsolationMVArun2v1Medium.mapping[0].cut = cms.string("RecoTauTag_tau
process.rerunDiscriminationByIsolationMVArun2v1Tight = process.rerunDiscriminationByIsolationMVAr
process.rerunDiscriminationByIsolationMVArun2v1Tight.mapping[0].cut = cms.string("RecoTauTag_tauI
process.rerunDiscriminationByIsolationMVArun2v1VTight = process.rerunDiscriminationByIsolationMVA
process.rerunDiscriminationByIsolationMVArun2v1VTight.mapping[0].cut = cms.string("RecoTauTag_tau
process.rerunDiscriminationByIsolationMVArun2v1VVTight = process.rerunDiscriminationByIsolationMV
process.rerunDiscriminationByIsolationMVArun2v1VVTight.mapping[0].cut = cms.string("RecoTauTag_ta

# this sequence has to be included in your cms.Path() before your analyzer which accesses the new
process.rerunMvaIsolation2SeqRun2 = cms.Sequence(
   process.rerunDiscriminationByIsolationMVArun2v1raw
```

```
   *process.rerunDiscriminationByIsolationMVArun2v1VLoose
   *process.rerunDiscriminationByIsolationMVArun2v1VVLoose
   *process.rerunDiscriminationByIsolationMVArun2v1Loose
   *process.rerunDiscriminationByIsolationMVArun2v1Medium
   *process.rerunDiscriminationByIsolationMVArun2v1Tight
   *process.rerunDiscriminationByIsolationMVArun2v1VTight
   *process.rerunDiscriminationByIsolationMVArun2v1VVTight
)

# 2017v2 dR=0.3
self.tauIdDiscrMVA_2017_version = "v2"
self.tauIdDiscrMVA_trainings_run2_2017 = {
   'tauIdMVAIsoDBoldDMdR0p3wLT2017' : "tauIdMVAIsoDBoldDMdR0p3wLT2017",
}
self.tauIdDiscrMVA_WPs_run2_2017 = {
   'tauIdMVAIsoDBoldDMdR0p3wLT2017' : {
      'Eff95' : "DBoldDMdR0p3wLTEff95",
      'Eff90' : "DBoldDMdR0p3wLTEff90",
      'Eff80' : "DBoldDMdR0p3wLTEff80",
      'Eff70' : "DBoldDMdR0p3wLTEff70",
      'Eff60' : "DBoldDMdR0p3wLTEff60",
      'Eff50' : "DBoldDMdR0p3wLTEff50",
      'Eff40' : "DBoldDMdR0p3wLTEff40"
   }
}
self.loadMVA_WPs_run2_2017()

process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2raw = patDiscriminationByIsolationMVA
   PATTauProducer = cms.InputTag('slimmedTaus'),
   Prediscriminants = noPrediscriminants,
   loadMVAfromDB = cms.bool(True),
   mvaName = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMdR0p3wLT2017v2"),
   mvaOpt = cms.string("DBoldDMwLTwGJ"),
   requireDecayMode = cms.bool(True),
   verbosity = cms.int32(0)
)

process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VLoose = patDiscriminationByIsolation
   PATTauProducer = cms.InputTag('slimmedTaus'),
   Prediscriminants = noPrediscriminants,
   toMultiplex = cms.InputTag('rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2raw'),
   key = cms.InputTag('rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2raw:category'),
   loadMVAfromDB = cms.bool(True),
   mvaOutput_normalization = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMdR0p3wLT2017v2_mvaOutput_no
   mapping = cms.VPSet(
      cms.PSet(
         category = cms.uint32(0),
         cut = cms.string("RecoTauTag_tauIdMVAIsoDBoldDMdR0p3wLT2017v2_WPEff90"),
         variable = cms.string("pt"),
      )
   ),
   verbosity = cms.int32(0)
)

process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VVLoose = process.rerunDiscrimination
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VVLoose.mapping[0].cut = cms.string("
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Loose = process.rerunDiscriminationBy
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Loose.mapping[0].cut = cms.string("Re
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Medium = process.rerunDiscriminationB
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Medium.mapping[0].cut = cms.string("R
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Tight = process.rerunDiscriminationBy
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Tight.mapping[0].cut = cms.string("Re
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VTight = process.rerunDiscriminationB
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VTight.mapping[0].cut = cms.string("R
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VVTight = process.rerunDiscrimination
process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VVTight.mapping[0].cut = cms.string("
```

```
process.rerunMvaIsolationSequence += cms.Sequence(
   process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2raw
   *process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VLoose
   *process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VVLoose
   *process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Loose
   *process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Medium
   *process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2Tight
   *process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VTight
   *process.rerunDiscriminationByIsolationOldDMdR0p3MVArun2017v2VVTight
)


# embed new id's into new tau collection
embedID = cms.EDProducer("PATTauIDEmbedder",
   src = cms.InputTag('slimmedTaus'),
   tauIDSources = cms.PSet(
      byIsolationMVArun2v1DBoldDMwLTrawNew = cms.InputTag('rerunDiscriminationByIsolationMVArun2v
      byVLooseIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAru
      byVVLooseIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAr
      byLooseIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVArun
      byMediumIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAru
      byTightIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVArun
      byVTightIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAru
      byVVTightIsolationMVArun2v1DBoldDMwLTNew = cms.InputTag('rerunDiscriminationByIsolationMVAr
      byIsolationMVArun2017v2DBoldDMdR0p3wLTraw2017 = cms.InputTag('rerunDiscriminationByIsolatio
      byVVLooseIsolationMVArun2017v2DBoldDMdR0p3wLT2017 = cms.InputTag('rerunDiscriminationByIsol
      byVLooseIsolationMVArun2017v2DBoldDMdR0p3wLT2017 = cms.InputTag('rerunDiscriminationByIsola
      byLooseIsolationMVArun2017v2DBoldDMdR0p3wLT2017 = cms.InputTag('rerunDiscriminationByIsolat
      byMediumIsolationMVArun2017v2DBoldDMdR0p3wLT2017 = cms.InputTag('rerunDiscriminationByIsola
      byTightIsolationMVArun2017v2DBoldDMdR0p3wLT2017 = cms.InputTag('rerunDiscriminationByIsolat
      byVTightIsolationMVArun2017v2DBoldDMdR0p3wLT2017 = cms.InputTag('rerunDiscriminationByIsola
      byVVTightIsolationMVArun2017v2DBoldDMdR0p3wLT2017 = cms.InputTag('rerunDiscriminationByIsol
   ),
   )

setattr(process, "NewTauIDsEmbedded", embedID)

# inclusion in the process
process.p += process.rerunMvaIsolation2SeqRun2
process.p += getattr(process, "NewTauIDsEmbedded")
#  then you can continue with your ntuple creation process for example
```

Once the new discriminators are embedded into the new tau collection (in this example called "NewTauIDsEmbedded") one can simply retrieve them via the `pat::Tau::tauID` function in a loop over the collection like so:

```
float newIsolationMVArawValue = tau->tauID("byIsolationMVArun2v1DBoldDMwLTrawNew");
```

Running of the DNN-based tau ID on MiniAOD event content with CMSSW 9_4_X / 10_2_X / 10_4_X (experimental) NEW

While it is not possible to fully rebuild taus from PF particles with MiniAOD event content, it is possible to (re)calculate tau-Id variables with both "classic" BDT-based algorithms as well as with new experimental deep-neural-network-based (DNN-based) ones. This section provides a recipe on how to get those tau-Ids.

DNN-based Tau-Ids which are accessible thanks to this recipe

Tau-Id	Name in pat::Tau accessed via tau.tauID(name)	Notes
DeepTau vs jets	*byDeepTau2017v1VSjetraw*	Raw DNN score

	by[WP]DeepTau2017v1VSjet	WP=VVVLoose,VVLoose,VLoose,Loose,Medium,Tight,VTight,VVTight
DeepTau vs e	*byDeepTau2017v1VSeraw*	Raw DNN score
	by[WP]DeepTau2017v1VSe	WP=VVVLoose,VVLoose,VLoose,Loose,Medium,Tight,VTight,VVTight
DeepTau vs μ	*byDeepTau2017v1VSmuraw*	Raw DNN score
	by[WP]DeepTau2017v1VSmu	WP=VVVLoose,VVLoose,VLoose,Loose,Medium,Tight,VTight,VVTight
DPFTau vs all (v0)	*byDpfTau2016v0VSallraw*	Raw DNN score, mostly against jets, some power against e
	byTightDpfTau2016v0VSall	Tight WP, the only one defined
DPFTau vs all (v1)	*byDpfTau2016v1VSallraw*	Raw DNN score, mostly against jets, some power against e
	byTightDpfTau2016v1VSall	Tight WP, dummy cuts

Note: also "classic" MVAIso 2017v1/v2 can be added with the ***runTauIdMVA.py*** tool discussed below.

1. Setup CMSSW area. DNN-based tau ids are available in the follwing CMSSW releases 9_4_X (where X >= 13), 10_2_X (where X >= 9), and any release >= 10_4_0.

2. Add new tau_Ids to your CMSSW python configuration using the ***runTauIdMVA.py*** tool:

```
[...]
updatedTauName = "slimmedTausNewID" #name of pat::Tau collection with new tau-Ids
import RecoTauTag.RecoTau.tools.runTauIdMVA as tauIdConfig
tauIdEmbedder = tauIdConfig.TauIDEmbedder(process, cms, debug = False,
                    updatedTauName = updatedTauName,
                    toKeep = [ "2017v2", "dR0p32017v2", "newDM2017v2", #classic MVAIso tau-Ids
                               "deepTau2017v1", #deepTau Tau-Ids
                               "DPFTau_2016_v0", #D[eep]PF[low] Tau-Id
                               ])
tauIdEmbedder.runTauID()
# Path and EndPath definitions
process.p = cms.Path(
    process.rerunMvaIsolationSequence *
    getattr(process,updatedTauName)
)
[...]
```

Note 1: Do not forget either store the ***updatedTauName*** collection to output file or read it in your ntuplizer.
Note 2: You can check configuration details in the example configuration file:
$CMSSW_RELEASE_BASE/src/RecoTauTag/RecoTau/test/runDeepTauIDsOnMiniAOD.py

Optional If you want to try to run on CRAB a new NN which is not yet integrated into the official CMSSW releases and its size is too big to fit into CRAB sendbox, you should:

Show details Hide details
a) Remove the files from your local CMSSW area:

```
rm -rf RecoTauTag/TrainingFiles/data
```

This will prevent you from running the DNN-based tau-Ids locally. b) Add the ***get training files*** step to a script executed by Crab following this advice: Running a user script with CRAB. The script (*myscript.sh*) should looks as follows:

```
# Get DNN training files
git clone https://github.com/cms-tau-pog/RecoTauTag-TrainingFiles -b master RecoTauTag/TrainingFi
# Execute the CMSSW job:
```

```
cmsRun -j FrameworkJobReport.xml -p PSet.py
```

and it has to be added to Crab configuration

```
config.JobType.scriptExe = 'myscript.sh'
```

Running of the DeepTauIDs ver. 2017v2p1 with 10_2_16/10_2_16_UL (or newer) NEW

Note DeepTauIDs ver. 2017v2p1 is ver. 2017v2 with disabled usage of dxy_PCA coordinates which fixes big data-MC discrepancy.
Note CMSSW_10_2_16_UL is available at CC7 machines (e.g. lxplus), while CMSSW_10_2_16 at SLC6 ones (e.g. lxplus6),

1. Setup CMSSW area:

```
# Setup CMSSW area
cmsrel CMSSW_10_2_16_UL
cd CMSSW_10_2_16_UL/src/
cmsenv
# Update DeepTau code and store DeepTauIDs in nanoAOD by a checkout from Tau POG repository
# (note "-u" option preventing checkout of unnecessary stuff)
git cms-merge-topic -u cms-tau-pog:CMSSW_10_2_X_tau-pog_DeepTau2017v2p1_nanoAOD
# No needed to checkout training files for CMSSW>=10_2_16/10_2_16_UL
# Compile
scram b -j 4
```

2. (for miniAOD users) Add new DeepTauIDs to your CMSSW python configuration using the ***runTauIdMVA.py*** tool like this:

```
[...]
updatedTauName = "slimmedTausNewID" #name of pat::Tau collection with new tau-Ids
import RecoTauTag.RecoTau.tools.runTauIdMVA as tauIdConfig
tauIdEmbedder = tauIdConfig.TauIDEmbedder(process, cms, debug = False,
                    updatedTauName = updatedTauName,
                    toKeep = ["deepTau2017v2p1", #deepTau TauIDs
                              ])
tauIdEmbedder.runTauID()
# Path and EndPath definitions
process.p = cms.Path(
    process.rerunMvaIsolationSequence *
    getattr(process,updatedTauName)
)
[...]
```

Note 1: Do not forget either store the ***updatedTauName*** collection to output file or read it in your ntuplizer.
Note 2: You can check configuration details in the example configuration file:
$CMSSW_RELEASE_BASE/src/RecoTauTag/RecoTau/test/runDeepTauIDsOnMiniAOD.py
Note 3: To run DeepTauIDs 2017v2 (w/o fix) add ***deepTau2017v2*** to ***toKeep***.
Note 4: Old version of DeepTauIDs (2017v1) and DPFTauIDs are not supported with this setup due to missing training files. Users who wants run older versions of DNN-based TauIDs in parallel should checkout ***DeepTau2017v2*** branch of ***RecoTauTag-TrainingFiles***. However, to run all those TauIDs with Crab one should apply the trick mentioned above (due to limited size of sandbox).

Private NanoAOD production with DeepTauIDs

After the cms-merge-topic indicated in the installation instructions, the DeepTau IDs (v2p1) are included into the NanoAOD sequence (link, link), and the variables listed in the table below are be stored in the NanoAOD root-tuple.

To get NanoAOD configuration one can execute following cmsDriver command (or similar depending on era):

```
cmsDriver.py NanoAODv5_deepTauID --filein file:miniAOD2017v2.root --fileout file:nanoAODv5_deepTa
```

Following DeepTau IDs are accessible with this recipe:

Tau-Id	Name in pat::Tau accessed via tau.tauID(name)	Name in NanoAOD	Notes
DeepTau vs jets	*byDeepTau2017v2p1VSjetraw*	rawDeepTau2017v2p1VSjet	Raw DNN score
	by[WP]DeepTau2017v2p1VSjet	idDeepTau2017v2p1VSjet	WP=VVVLoose,VVLoose,VLoose,Loose,M
DeepTau vs e	*byDeepTau2017v2p1VSeraw*	rawDeepTau2017v2p1VSe	Raw DNN score
	by[WP]DeepTau2017v2p1VSe	idDeepTau2017v2p1VSe	WP=VVVLoose,VVLoose,VLoose,Loose,M
DeepTau vs μ	*byDeepTau2017v2p1VSmuraw*	rawDeepTau2017v2p1VSmu	Raw DNN score
	by[WP]DeepTau2017v2p1VSmu	idDeepTau2017v2p1VSmu	WP=VLoose,Loose,Medium,Tight

Working points against a given source of fakes are measured requiring the loosest working point against other sources (e.g. to measure VSjet WPs, VVVLoose VSe and VLoose VSmu were required). This procedure excludes the part of the parameter space which is very far from the tau signal region, where the MC modeling can be less accurate and, also the correlations between VS e, mu, jet are stronger. The loosest working points have very high efficiency for the real taus, so it is recommended to always apply them, i.e. to use only tau candidates that pass at least VVVLoose VSe, VLoose VSmu and VVVLoose VSjet.

How to get latest & greatest tau ID ? NEW

The simplest way is to run the cmsDriver command for MiniAOD production (starting from AOD or whatever), with release where the desired implementation was introduced (or later version).

For example, the latest implementations such as anti-e MVA, MVA isolation, removal of 3prong + pi0 etc are introduced with 7_4_12 (MiniAODv2 with 74X) or 7_6_2 (MiniAODv2 with 76X).

```
cmsDriver.py miniAOD-prod -s PAT --eventcontent MINIAODSIM --runUnscheduled --mc --filein /store/
```

As long as you just read RelVals global tag is not very important as you do not read any conditions, so you can safely use automatic one, i.e. auto:run2_mc.

By running the configuration file, you can get the MiniAOD with latest greatest tauID.

Guidelines for developing the tau POG code NEW

Guidelines for Tau POG developers can be found on the following TWiki page: SWGuidePFTauIDDevelopers.

Tau ID sequences

Legacy Tau ID (Run I)

This ID was the recommendation for all the analyses using Run I data and corresponding MC. It has been used in all tau-related publications related to Run I data. Since 2014 it is being replaced by a new and improved algorithm (see below)

72X and newer

Show Hide
The legacy tau ID is not supported any more in CMSSW_7_2_X release series

71X

Show Hide
The legacy tau ID has been present until CMSSW_7_1_0_pre1. Since CMSSW_7_1_0_pre2 it has been superseded by the new tau ID for Run II (see below).

70X

Show Hide
The legacy tau ID has been present until CMSSW_7_0_0_pre12. Since CMSSW_7_0_0_pre13 it has been superseded by the new tau ID for Run II (see below).

620_SLHCX

Show Hide
Spring 2013 discriminators and performance improvements (equivalent of RecoTau V01-04-25 + Configuration V01-04-13) are available out-of-the box in CMSSW_6_2_0_SLHC9 and older releases

62X

Show Hide
Spring 2013 discriminators and performance improvements (equivalent of RecoTau V01-04-25 + Configuration V01-04-13) are available out-of-the box in CMSSW_6_2_0_pre6 and newer releases

61X

Show Hide
For the Phase1 and Phase2 studies, Legacy tau ID has been backported to CMSSW_6_1_2_SLHC4_patch2. To profit from it, do

```
cmsrel CMSSW_6_1_2_SLHC4_patch2
cd CMSSW_6_1_2_SLHC4_patch2/src
cmsenv
git cms-merge-topic -u cms-tau-pog:CMSSW_6_1_2_SLHC4_patch2_tauLegacy2012
```

Afterwards, you should rerun the tau sequence:

- add `process.load("RecoTauTag.Configuration.RecoPFTauTag_cff")` to your cfg
- add `process.PFTau` to your path

53X

Show Hide
The legacy tau ID software contains several discriminants introduced in the end of 2012 and in 2013 that were not run during the official simulation and data reconstruction (see Full list of recommended discriminants .

In order to obtain recommended tau objects, it is necessary to re-run full HPS sequence:

```
process.load("RecoTauTag.Configuration.RecoPFTauTag_cff")
```

and add `process.recoTauClassicHPSSequence` to your path.

The configuration has been tested to work with PF2PAT and latest PAT recommendations as listed SWGuidePATReleaseNotes52X. More details about this recommendation available in the talk

You should also add following lines to your config in order to use the latest discriminators:

```
from PhysicsTools.PatAlgos.tools.tauTools import *
switchToPFTauHPS(process)
```

Changes wrt. recommendation presented in this talk:

- Included implementation for AntiMu3 as presented in this talk
- bugfix in AgainstMuonTight discriminator (slide 10 of this talk)
- Alternative tau 4-vector is calculated as a sum of all PFcands in a cone 0.12

Please, have a look at TauIDRecommendation to see recommendations on how to use the new discriminants.

5_3_12 and higher

The tau ID recommended for analysis of 2011-2012 LHC data is included in CMSSW_5_3_12. In order to benefit from it, you need to do just

```
cmsrel CMSSW_5_3_12
cd CMSSW_5_3_12/src
cmsenv
```

A nutshell recipe for samples reconstructed with CMSSW>=5_3_12 (click here)

Do not forget to rerun recoTauClassicHPSSequence (see above) as tau ID stored in AODs produced by CMSSW < 5_3_12 is outdated

CMSSW < 5_3_12

via GitHub (recommended version)

Show Hide
Include the central package and update your branch:

```
cmsrel CMSSW_5_3_11_patch6
cd CMSSW_5_3_11_patch6/src
cmsenv
git cms-addpkg RecoTauTag/RecoTau
git cms-merge-topic -u cms-tau-pog:CMSSW_5_3_X
```

If you wish to add new PAT features, please follow the recipes on PAT SW guide. Use only the recipes for the git transition and newer (on the top of the page).

This recipe has been tested to work on release CMSSW_5_3_11_patch6 and it is possible that some merging problems might appear in older releases. Write to hn-cms-tauid@cernNOSPAMPLEASE.ch in case of any troubles.

via CVS (not recommended)

CVS is not updated since CMSSW_5_3_9_patch3

Show Hide
Checkout the PAT recommendations (version V08-09-57 or higher) and then

```
cmsrel CMSSW_5_3_9
cd CMSSW_5_3_9/src
cmsenv
cvs co -r V01-04-25 RecoTauTag/RecoTau #HCP + new discriminants
cvs co -r V01-04-13 RecoTauTag/Configuration
```

If you are using some 52X release, you also need to do

```
cvs co -r V00-04-00 CondFormats/EgammaObjects # not necessary if CMSSW >= 5_3_0
```

4XX

Show Hide
Recipe to get the same taus with the same discriminators as are provided for the 53x and 6xx releases above. See SWGuidePATReleaseNotes42X and SWGuidePATReleaseNotes44X for full list of recommended PAT tags and on top of those recommendations do:

via CVS (not supported any more)

```
export CVSROOT=":ext:<cern-user-account>@lxplus5.cern.ch:/afs/cern.ch/user/c/cvscmssw/public/CMSS
cvs co -r CMSSW_5_2_4 DataFormats/TauReco  # yes, this is correct
cvs co -r CMSSW_5_2_4 RecoTauTag/TauTagTools
cvs co -r V01-04-25-4XX RecoTauTag/RecoTau #Legacy tau ID adapted for 4xx releases
cvs co -r V01-04-12-4XX RecoTauTag/Configuration
cvs co -r V00-04-00 CondFormats/EgammaObjects
cvs co PhysicsTools/IsolationAlgos # You need to recompile PAT packages which depend on DataForma
cvs co -r V08-07-53 PhysicsTools/PatAlgos    # or higher. See https://twiki.cern.ch/twiki/bin/vie
cvs co  DataFormats/PatCandidates
scram b -j 9
```

Git based recipe: It is possible to obtain the recipe above also using git:

```
cmsrel CMSSW_4_4_5_patch5 #or higher
cd CMSSW_4_4_5_patch5/src
cmsenv
git cms-addpkg RecoTauTag/RecoTau
git cms-merge-topic -u cms-tau-pog:CMSSW_4_4_X
```

If everything compiles, you should re-run the PFTau sequence

Afterwards, you should rerun the tau sequence:

- add `process.load("RecoTauTag.Configuration.RecoPFTauTag_cff")` to your cfg
- add `process.PFTau` to your path

Tau ID 2014 (preparation for Run II)

This algorithm is the state-of-the-art tau ID that has been developed in 2013 and included in official CMSSW releases since 2014. The main features of are improvement of the high pt (500-1000 GeV) tau reconstruction (efficiency, momentum reconstruction), tools for the reconstruction of taus produced in decays of boosted particles (reconstruction of taus using subjets) and exploitation of the tau lifetime variables (impact parameter and secondary vertex) which improves the tau performance. More details in this talk

To obtain the code:

80X

Show Hide
The new dynamic strip reconstruction is included by default in CMSSW_7_4_14. Use this or higher versions of CMSSW to reconstruct taus with dynamic strip reconstruction. Also included is the SWGuideBoostedTauID, please click the link for more details. Please switch to MiniAOD samples produced with CMSSW_8_0_10 as soon as they are ready.

76X

Show Hide
Use this or higher versions of CMSSW to reconstruct taus with dynamic strip reconstruction.

74X

Show Hide
This version is superseded by 76X for 2015 analysis. The new dynamic strip reconstruction is included by default in CMSSW_7_4_14. Use this or higher versions of CMSSW to reconstruct taus with dynamic strip reconstruction. Please switch to MiniAOD samples produced with CMSSW_7_4_14 as soon as they are ready.

72X

Show Hide
The 2014 tau ID is present since the first pre-release. The 72x is the current development release, so if you want to profit from continuous tau developments, use the latest available pre-release and re-run the tau ID.

Other old releases

Show Hide

71X

The CMSSW_7_1_X is current latest production release. Therefore it is closed for most developments (including taus). In order to profit from the latest developments, you can backport from 72x via

```
cmsenv
git cms-merge-topic -u cms-tau-pog:CMSSW_7_1_X_taus
```

70X

The CMSSW_7_0_X was the main production release until end of June 2014. Therefore it is closed for most developments (including taus). In order to profit from the latest developments, you can backport from 72x via

```
cmsenv
git cms-merge-topic -u cms-tau-pog:CMSSW_7_0_X_taus
```

62X

The 2014 tau algorithm can be included in CMSSW_6_2_X series. The code can be obtained from cms-tau-pog github repository:

```
cmsenv
git cms-merge-topic -u cms-tau-pog:CMSSW_6_2_X_HighPt
```

This software branch does not contain yet the implementation of tau reconstruction in subjets and analyses in this release are not supported by the tau POG.

61X

The 2014 tau algorithm is not supported in CMSSW_6_1_X series.

53X (recommendation for Run I analyses)

Show Hide
Recommendation: The 2014 tau ID has been developed and tested primarily in 53x releases so it is useful for the analyses using Run I data where boosted taus can appear. Also, due to the exploitation of the lifetime variables, it brings improvement also in the low pt (Z, H) range. The code can be obtained from github repository:

```
cmsrel CMSSW_5_3_13_patch3 #or any CMSSW_5_3_x release >= 5_3_13
cd CMSSW_5_3_13_patch3/src
cmsenv
# git cms-merge-topic -u cms-tau-pog:CMSSW_5_3_X_boostedTaus_2013Dec17 # old recommendation
git cms-merge-topic -u cms-tau-pog:CMSSW_5_3_X_tauID2014 # new recommendation with the same perfo
```

This commands will provide you with the head of CMSSW_5_3_X_tauID2014 branch and merge it with your release (and whatever else you have in the working area). Note that current version of code has been tested to work in CMSSW_5_3_13, CMSSW_5_3_14 and CMSSW_5_3_15. The

If everything compiles, you should rerun the tau sequence:

- add `process.load("RecoTauTag.Configuration.RecoPFTauTag_cff")` to your cfg
- add `process.PFTau` to your path

Note: The new high Pt tau reconstruction tag comes with new discriminators:

- MVA5 anti-e discriminator: This discriminator replaces the MVA3 anti-e discriminator
- MVA anti-mu discriminator: This is a new discriminator against muons. According to MC it reduces the mu -> tau fake-rate by a factor 3-4 compared to the anti-mu3 discriminator (for the same efficiency). The MVA anti-mu discriminator is however not validated with data yet.
- MVA tau ID discriminators: These come in 2 variants, including ("wLT") and not including ("woLT") tau transverse impact parameter and decay vertex information as MVA input variables, in addition to isolation Pt sums. There exists two sets of discriminators, trained on 1-prong and 3-prong tau candidates ("oldDM") and trained on 1-prong, "2-prong" and 3-prong tau candidates ("newDM").

It is recommended to keep all discriminators in your PAT-tuples/Ntuples, so that you have the flexibility to switch between cut based and MVA based discriminators and between different working-points. In order to use these discriminators (and avoid crash when searching for deprecated ones) you should add following lines to your config:

```
from PhysicsTools.PatAlgos.tools.tauTools import *
switchToPFTauHPS(process)
```

Note: The tau data-formats have changed in the new high Pt tau reconstruction tag. This means that you need to drop all tau related information "on input" when reading AOD/RECO files. To do that, please add the following lines to the python config of your PoolSource module:

```
process.source = cms.Source("PoolSource",
    fileNames = cms.untracked.vstring(
        ...
    ),
```

```
    dropDescendantsOfDroppedBranches = cms.untracked.bool(False),
    inputCommands = cms.untracked.vstring(
        'keep *',
        'drop recoPFTaus_*_*_*'
    )
)
```

Developer's version

Show Hide
The latest version of the code, where we develop and test new features, can be obtained in the similar way:

```
cmsrel CMSSW_5_3_15 #or any CMSSW_5_3_x release >= 5_3_15
cd CMSSW_5_3_15/src
cmsenv
git cms-merge-topic -u cms-tau-pog:CMSSW_5_3_X_tauID2014 # branch with the latest version of the
```

Please, do not use CMSSW_5_3_X_boostedTaus branch for any further developments. It has been deprecated and will be archived soon.

To develop the code as one of the developers of cms-tau-pog git organization (**PLEASE USE THE PULL REQUEST MECHANISM DESCRIBED BELOW**):

Show Hide

```
git remote add tau-pog git@github.com:cms-tau-pog/cmssw.git # add new remote
git fetch tau-pog # read from remote (get the list of branches and tags)
git checkout -t remotes/tau-pog/CMSSW_5_3_X_tauID2014 # make a new local branch called CMSSW_5_3_
```

When you are happy with local developments (using `git add/rm` and `git commit`), the command `git push` will directly update `CMSSW_5_3_X_tauID2014` branch in cms-tau-pog repository.

To develop as any cmssw user:

Show Hide

```
git checkout -b TauHighPtTestFeature cms-tau-pog/CMSSW_5_3_X_tauID2014 # make a local branch TauH
```

Do a local developments and commit to local git repository (using `git add/rm` and `git commit`) and then, if you want, you can push the whole branch to your fork of cmmsw:

```
git push -u my-cmssw TauHighPtTestFeature
```

If you want to have your changes included in "official" cms-tau-pog repository, use a pull request (make sure you are pulling to cms-tau-pog and not to cmssw-official)

Tau ID Algorithms

The recommended tau ID algorithm is based on ParticleFlow and is called HPS).

The tau reconstruction code can be found in:

- DataFormats/TauReco - Definition of the dataformats (see event content)
- RecoTauTag/RecoTau - Implementation of the discriminators and decay mode finder.

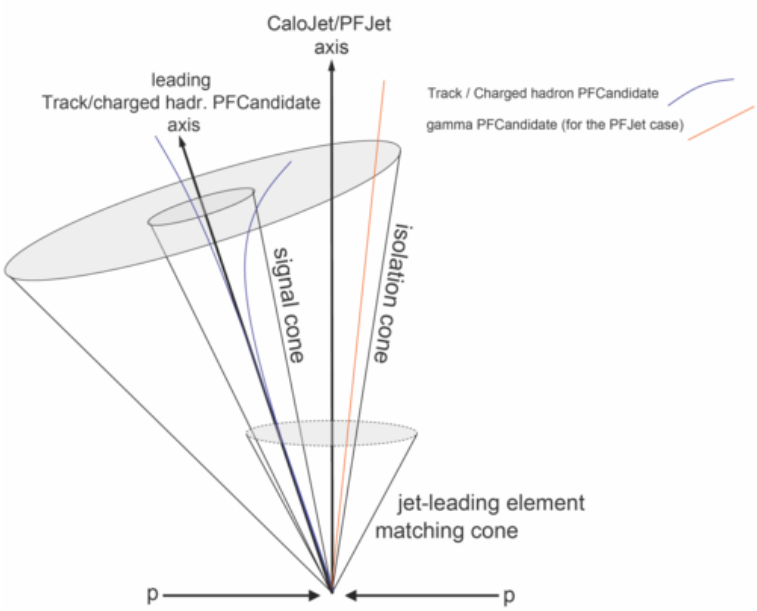
Creation

Rerun the tau sequences, load the tau steering sequence in your cfg file:
process.load("RecoTauTag.Configuration.RecoPFTauTag_cff")

On RECO level reco::PFTaus are used to store all possible taus jets. The result of the various discriminators is stored in reco::PFTauDiscriminator.

The initial PFTaus collection will first contain one tau candidate for each reconstructed PFJet (using the anti-kt algorithm with a cone size of 0.5). The jet direction is set to the direction of the leading charged hadron in the jet (highest pT), which should be within $\Delta R = 0.1$ with respect to the jet axis. In this stage the signal- and isolation cones as well as the decay mode are defined. Subsequently the various algorithms create PFTauDiscriminators which can be used to select a collection of PFTaus.

Explanation of signal- and isolation cones Hide



Usage

in your event loop (e.g. the **`analyze(const edm::Event& iEvent)`** method of your analyzer) do

Recipe Hide

```
// read PFTau collection (set tauSrc_ in to an edm::InputTag before)
    edm::Handle taus;
    iEvent.getByLabel(tauSrc_, taus);
//read one PFTauDiscriminator (set discriminatorSrc_ in to an edm::InputTag before)
    edm::Handle discriminator;
    iEvent.getByLabel(discriminatorSrc_, discriminator);
// loop over taus
    for ( unsigned iTau = 0; iTau < taus->size(); ++iTau ) {
        reco::PFTauRef tauCandidate(taus, iTau);
// check if tau candidate has passed discriminator
        if( (*discriminator)[tauCandidate] > 0.5 ){
        // do something with your candidate
        }
    }
```

Discriminators

The PFTauDiscriminators are used store the result of the various tau tagging algorithms and select jets that likely originate from a hadronic tau decay. By definition they are real numbers between 0 and 1 where higher numbers indicate a more positive outcome of the discriminator. Note that most discriminators are in fact binary thus taking just values of 0 (=did not pass) and 1 (=passed).

The names PFTauDiscriminator collections follow the **<algorithm-prefix>** + **<discriminator name>** convention. (e.g. the **AgainstElectron** collection can be accessed via **hpsPFTauDiscriminationAgainstElectron**)

Legacy tau ID (Run I)

The **<algorithm-prefix>** is **hpsPFTauDiscrimination**

Name	binary?	Description
ByLooseElectronRejection	yes	electron pion MVA discriminator < 0.6
ByMediumElectronRejection	yes	electron pion MVA discriminator < -0.1 and not 1.4442 <
ByTightElectronRejection	yes	electron pion MVA discriminator < -0.1 and not 1.4442 <
ByMVA3Loose/Medium/Tight/VTightElectronRejection	yes	anti-electron MVA discriminator with improved training (see talk)
ByMVA3VTightElectronRejection	yes	anti-electron MVA discriminator with improved training (same efficiency as "HCP 2012 working point" (see talk)
ByLooseMuonRejection	yes	Tau Lead Track not matched to chamber hits
ByMediumMuonRejection	yes	Tau Lead Track not matched to global/tracker muon
ByTightMuonRejection	yes	Tau Lead Track not matched to global/tracker muon and large enough energy deposit in ECAL + HCAL
ByLooseMuonRejection2	yes	Same as **AgainstMuonLoose**
ByMediumMuonRejection2	yes	Loose2 && no DT, CSC or RPC Hits in last 2 Stations
ByTightMuonRejection2	yes	Medium2 && large enough energy deposit in ECAL + HCAL in 1 prong + 0 strip decay mode ((ECAL+HCAL) > 0.2 * p_T)
ByDecayModeFinding	yes	You will always want to use this (see AN-10-82)
ByVLooseIsolation	yes	isolation cone of 0.3 , no PF Charged Candidates with pT > 1.5 GeV/c and no PF Gamma candidates with ET > 2.0 GeV
ByVLooseCombinedIsolationDBSumPtCorr	yes	isolation cone of 0.3 , Delta Beta corrected sum pT of PF charged and PF gamma isolation candidates (pT > 0.5 GeV) less than 3 GeV
ByLooseCombinedIsolationDBSumPtCorr	yes	isolation cone of 0.5 , Delta Beta corrected sum pT of PF charged and PF gamma

		isolation candidates (pT > 0.5 GeV) less than 2 GeV
ByMediumCombinedIsolationDBSumPtCorr	yes	isolation cone of 0.5 , Delta Beta corrected sum pT of PF charged and PF gamma isolation candidates (pT > 0.5 GeV) less than 1 GeV
ByTightCombinedIsolationDBSumPtCorr	yes	isolation cone of 0.5 , Delta Beta corrected sum pT of PF charged and PF gamma isolation candidates (pT > 0.5 GeV) less than 0.8 GeV
ByLooseCombinedIsolationDBSumPtCorr3Hits	yes	same as `ByLooseCombinedIsolationDBSumPtCorr` but requiring 3 hits (instead of 8) on track of isolation candidates
ByMediumCombinedIsolationDBSumPtCorr3Hits	yes	same as `ByMediumCombinedIsolationDBSumPtCorr` but requiring 3 hits (instead of 8) on track of isolation candidates
ByTightCombinedIsolationDBSumPtCorr3Hits	yes	same as `ByTightCombinedIsolationDBSumPtCorr` but requiring 3 hits (instead of 8) on track of isolation candidates
ByLooseIsolationMVA	yes	BDT based selection using isolation in rings around tau direction and shower shape variables
ByMediumIsolationMVA	yes	BDT based selection using isolation in rings around tau direction and shower shape variables
ByTightIsolationMVA	yes	BDT based selection using isolation in rings around tau direction and shower shape variables
ByIsolationMVAraw	no	output of BDT based selection using isolation in rings around tau direction and shower shape variables
ByLooseIsolationMVA2	yes	same as **ByLooseIsolationMVA** with new training and improved performance
ByMediumIsolationMVA2	yes	same as **ByMediumIsolationMVA** with new training and improved performance
ByTightIsolationMVA2	yes	same as **ByTightIsolationMVA** with new training and improved performance
ByIsolationMVA2raw	no	output of "MVA2" BDT discriminator

Isolation discriminators usage

Loosening of quality cuts on tracks of isolation candidates means that more of them can enter isolation pt. This results in slightly decreased efficiency of tau identification but also considerably decreased jet fake rate. More details in this talk.

The combined isolation is recommended for all taus, while MVA based isolation can provide better performance for low pt (< 100 GeV) taus. Since RecoTauTag /RecoTau V01-04-23 (V01-04-23-4XX-00 for 4XX analysis) a new training for MVA isolation is available. The discriminators have "IsolationMVA2" in their name.

"MVA3" anti-electron discriminators

The new MVA training provides working points with decreased electron fake rate when keeping the same efficiency as for previous training. All MVA3 discriminators veto tau candidates in a crack region between barrel and endcap, so they are **NOT RECOMMENDED** if you are using tau id for **TAU VETO** in your analysis.

"AgainstMuon2" discriminators

A drop of efficiency of anti-muon discriminator have been observed in high p_T. The fix is available in "Muon2" discriminators. More details in (talk 1 and talk 2)

HPS decay mode definition Hide

```
int A = tau.signalPFChargedHadrCands().size()
int B = tau.signalPFGammaCands().size()
```

Mode	A	B
one prong	1	0
one prong + pi0	1	>0
three prong	3	0

Tau ID 2014 (preparation for Run II)

The **<algorithm-prefix>** is **hpsPFTauDiscrimination**

Name	binary?	Descriptio
ByLooseElectronRejection	yes	electron pion MVA discrin
ByMediumElectronRejection	yes	electron pion MVA discrin and not 1.4442 <
ByTightElectronRejection	yes	electron pion MVA discrin and not 1.4442 <
ByMVA5(Loose/Medium/Tight/VTight)ElectronRejection	yes	anti-electron MVA discrim training
ByLooseMuonRejection	yes	Tau Lead Track not matche hits
ByMediumMuonRejection	yes	Tau Lead Track not matche global/tracker muon
ByTightMuonRejection	yes	Tau Lead Track not matche global/tracker muon and er ECAL + HCAL exceeding Track momentum
ByLooseMuonRejection3	yes	Tau Lead Track not matche one segment in muon syste deposit in ECAL + HCAL Lead Track momentum
ByTightMuonRejection3	yes	Tau Lead Track not matche one segment or hits in the stations of the muon syster in ECAL + HCAL at least Track momentum

ByMVA(Loose/Medium/Tight)MuonRejection	yes	BDT based anti-muon disc
ByMVArawMuonRejection	no	raw MVA output of BDT b discriminator
ByDecayModeFinding	yes	You will always want to us AN-10-82)
ByVLooseIsolation	yes	isolation cone of 0.3 , no P Candidates with pT > 1.5 G Gamma candidates with ET
ByVLooseCombinedIsolationDBSumPtCorr	yes	isolation cone of 0.3 , Delt sum pT of PF charged and isolation candidates (pT > than 3 GeV
ByLooseCombinedIsolationDBSumPtCorr	yes	isolation cone of 0.5 , Delt sum pT of PF charged and isolation candidates (pT > than 2 GeV
ByMediumCombinedIsolationDBSumPtCorr	yes	isolation cone of 0.5 , Delt sum pT of PF charged and isolation candidates (pT > than 1 GeV
ByTightCombinedIsolationDBSumPtCorr	yes	isolation cone of 0.5 , Delt sum pT of PF charged and isolation candidates (pT > than 0.8 GeV
ByLooseCombinedIsolationDBSumPtCorr3Hits	yes	same as ByLooseCombinedIsolati but requiring 3 hits (instead of isolation candidates
ByMediumCombinedIsolationDBSumPtCorr3Hits	yes	same as ByMediumCombinedIsolat but requiring 3 hits (instead of isolation candidates
ByTightCombinedIsolationDBSumPtCorr3Hits	yes	same as ByTightCombinedIsolati but requiring 3 hits (instead of isolation candidates
By(VLoose/Loose/Medium/Tight/VTight/VVTight)IsolationMVA3oldDMwoLT	yes	BDT based tau ID discrimi isolation Pt sums, trained c 3-prong tau candidates
ByIsolationMVA3oldDMwoLTraw	no	raw MVA output of BDT b discriminator based on isol trained on 1-prong and 3-p candidates
By(VLoose/Loose/Medium/Tight/VTight/VVTight)IsolationMVA3oldDMwLT	yes	BDT based tau ID discrimi isolation Pt sums plus tau l information, trained on 1-p 3-prong tau candidates
ByIsolationMVA3oldDMwLTraw	no	raw MVA output of BDT b discriminator based on isol plus tau lifetime informatio 1-prong and 3-prong tau ca
By(VLoose/Loose/Medium/Tight/VTight/VVTight)IsolationMVA3newDMwoLT	yes	BDT based tau ID discrimi isolation Pt sums, trained c

		"2-prong" and 3-prong tau
`ByIsolationMVA3newDMwoLTraw`	no	raw MVA output of BDT d discriminator based on isol trained on 1-prong, "2-pron tau candidates
`By(VLoose/Loose/Medium/Tight/VTight/VVTight)IsolationMVA3newDMwLT`	yes	BDT based tau ID discrimi isolation Pt sums plus tau l information, trained on 1-p and 3-prong tau candidates
`ByIsolationMVA3newDMwLTraw`	no	raw MVA output of BDT d discriminator based on isol plus tau lifetime informatio 1-prong, "2-prong" and 3-p candidates

Further Reading

- PAS PFT 08/001 CMS Strategies for tau reconstruction and identification using particle-flow techniques
- PAS PFT 09/001 Particle Flow Event Reconstruction in CMS and Performance for Jets, Taus, and Emiss
- AN PFT 10/207 Study of tau reconstruction algorithms using pp collisions data collected at √s = 7 TeV
- AN PFT 10/082 Prospects for measurement of (pp Z)·B(Z + −)withCMSinpp Collisions at √s = 7TeV(describing HPS)
- AN 2010/99 The Tau Neural Classifier algorithm: tau identification and decay mode reconstruction using neural networks
- J. Instrum. 7 (2012) P01001 Tau 2010 Commissioning paper, including description of algorithms

Main tauID tasks

- Tau Reconstruction Validation: [[https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideTauIDValidationInstructions][SWGuideTauIDVali

---

This topic: CMSPublic > SWGuidePFTauID
Topic revision: r192 - 2019-09-04 - KonstantinAndrosov