Outputs inspection CIFAR10

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```r
library("ggpubr")
```

```
## Warning: package 'ggpubr' was built under R version 4.0.5
```

```r
library(LDATS)
```

```
## Warning: package 'LDATS' was built under R version 4.0.5
```

```r
library(stringr)
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.0.3
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(reticulate)
```

```
## Warning: package 'reticulate' was built under R version 4.0.5
```

```
library(abind)
```

```
## Warning: package 'abind' was built under R version 4.0.3
```

```
library(ggVennDiagram)
```

```
## Warning: package 'ggVennDiagram' was built under R version 4.0.5
```

```
np <- import("numpy")

source("utils.R")
```

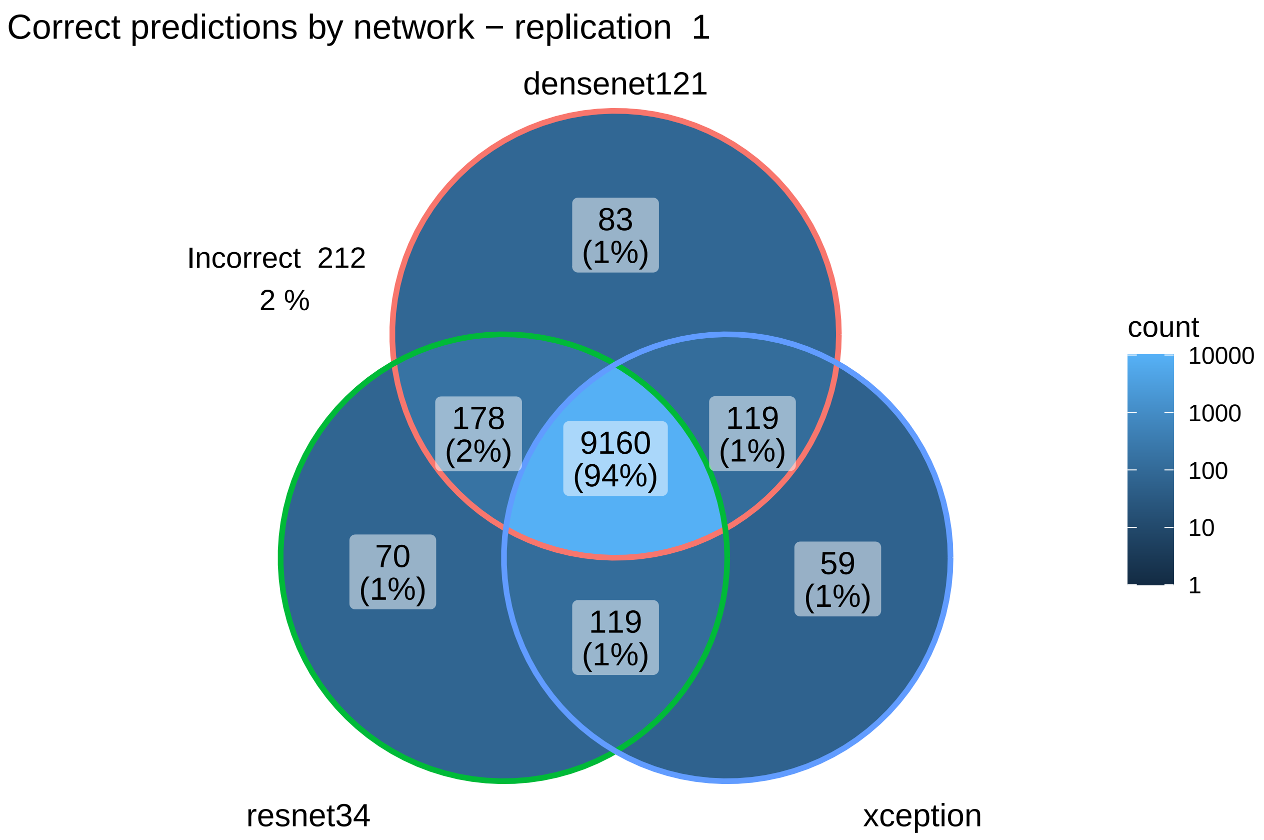
```
## Warning: package 'hash' was built under R version 4.0.5

## hash-2.2.6.1 provided by Decision Patterns

## Warning: package 'berryFunctions' was built under R version 4.0.5

##
## Attaching package: 'berryFunctions'

## The following object is masked from 'package:ggVennDiagram':
##
##     circle

## The following object is masked from 'package:dplyr':
##
##     between

## Warning: package 'purrr' was built under R version 4.0.3
```

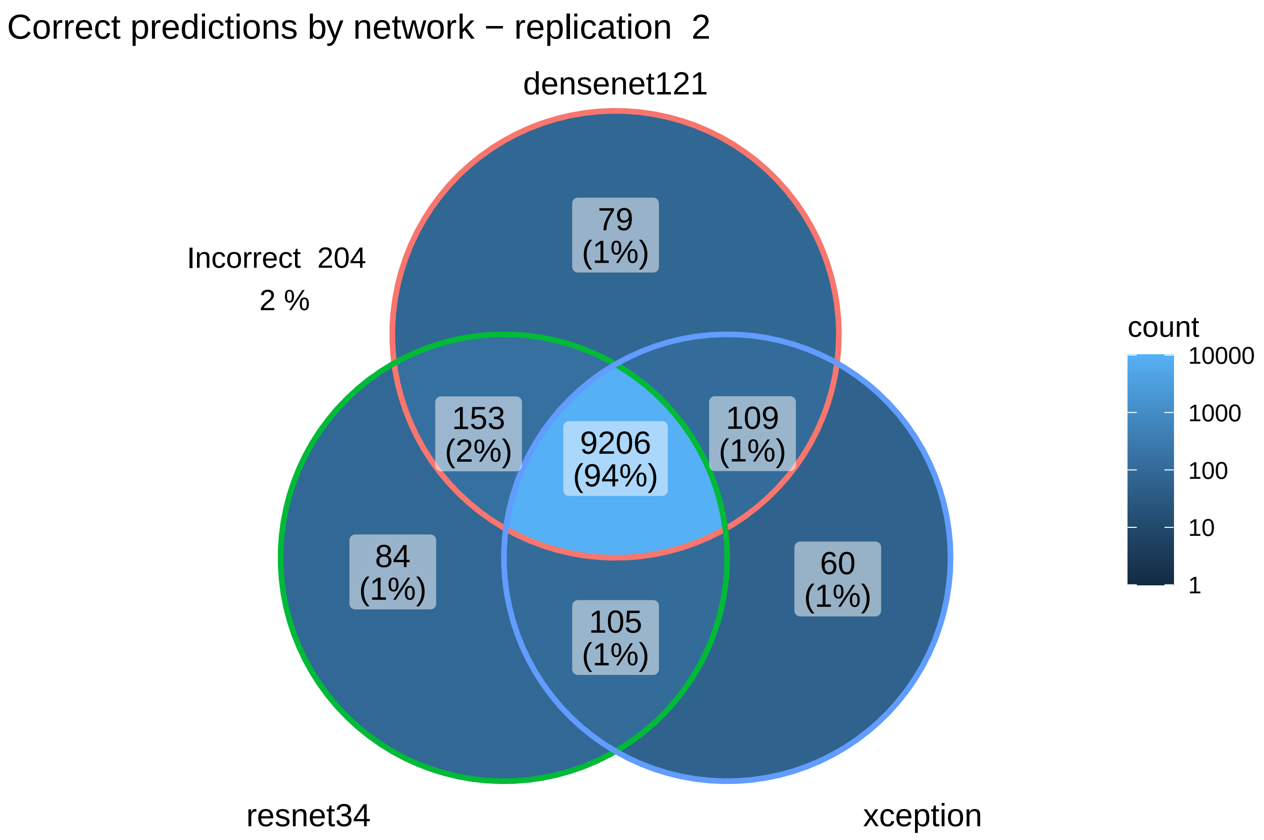
Visualization on CIFAR10. We are using data of three neural networks trained on reduced CIFAR10 training set. These networks were trained in 30 replications. In each replication, 500 samples from the training set were randomly extracted and formed validation set. In each replication, we trained two ensembles on the outputs of neural networks. First one was trained on randomly chosen subset, of size 500, of nn training set, second on the extracted validation set. In this visualization, we are trying to inspect the outputs deeper, mainly to make sense of strange behavior of nll metric for ensemble outputs.

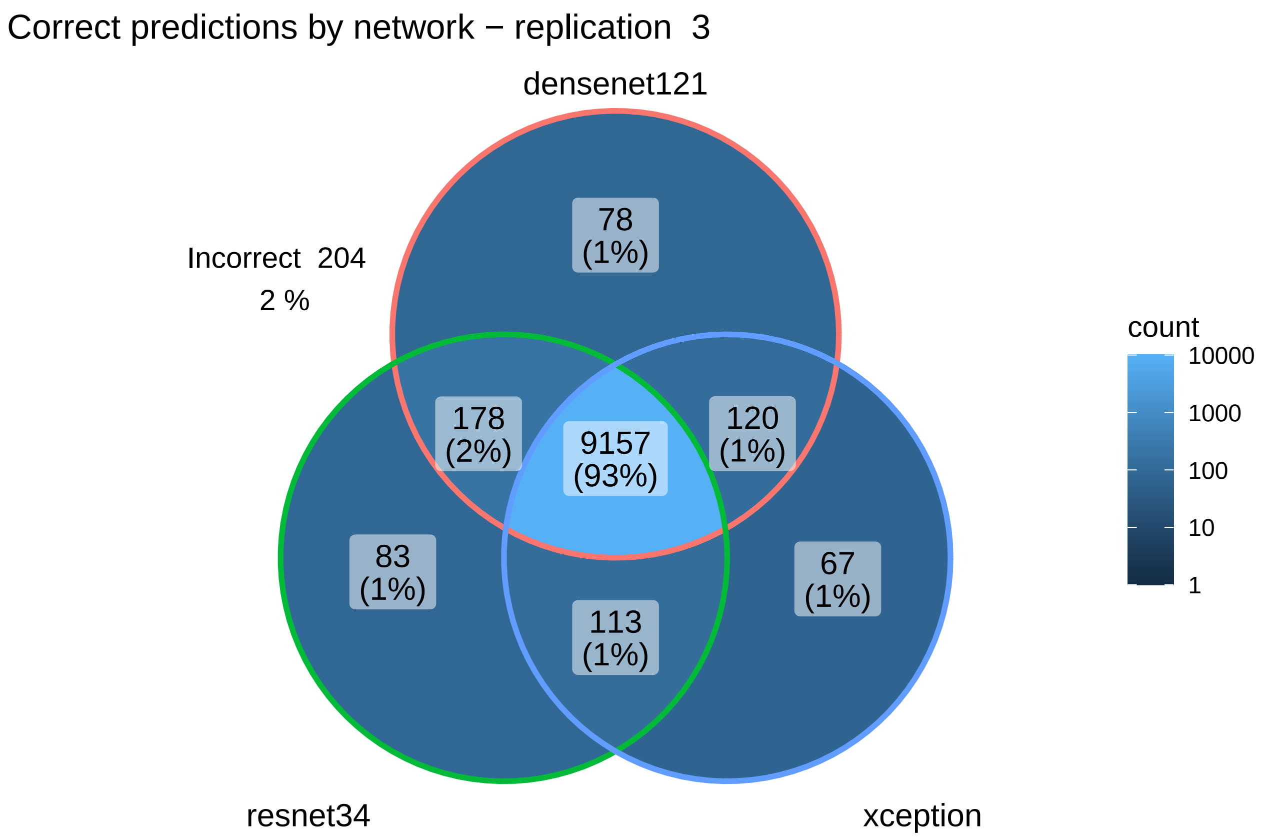
```
base_dir <- "../data/data_train_val_c10"
repls <- 0:29
classes <- 10

nets_outputs <- load_network_outputs(base_dir, repls)
ens_outputs <- load_ensemble_outputs(base_dir, repls)
net_results <- read.csv(file.path(base_dir, "net_accuracies.csv"))
ens_results <- read.csv(file.path(base_dir, "ensemble_accuracies.csv"))
```

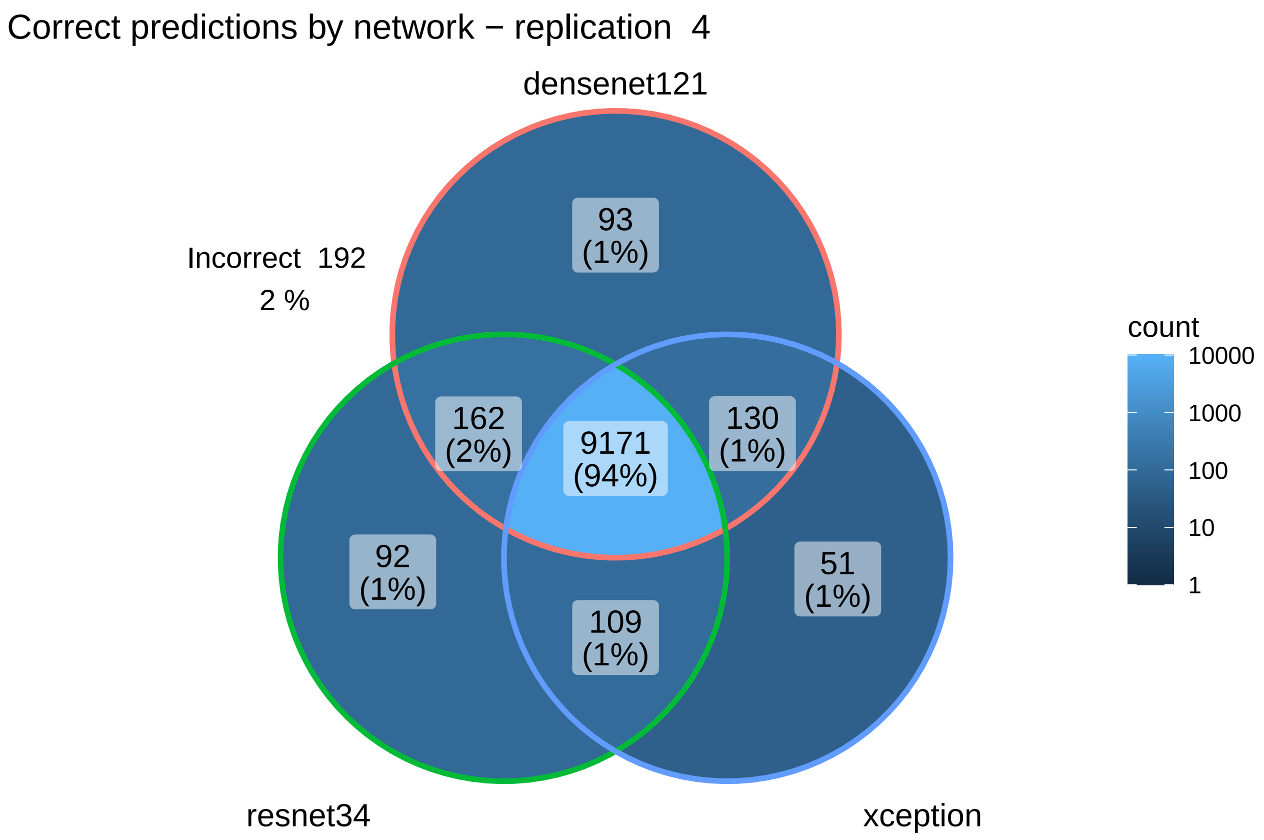
```
sort_ind <- function(lst)
{
  return(sort(lst, index.return=TRUE, decreasing=TRUE)$ix)
}
nets_test_top_indices <- apply(X=nets_outputs$test_outputs, MARGIN=c(1, 2, 3), FUN=sort_ind)[1, , , ]
r_n <- length(repls)
samples_n <- dim(nets_outputs$test_labels)[2]
nets_n <- length(nets_outputs$networks)
test_labs <- nets_outputs$test_labels + 1
dim(test_labs) <- c(r_n, 1, samples_n)
test_labs <- aperm(abind(array(rep(aperm(test_labs, perm=c(2, 1, 3)), nets_n), c(r_n, samples_n, nets_n
nets_test_cor_preds <- test_labs == nets_test_top_indices
```

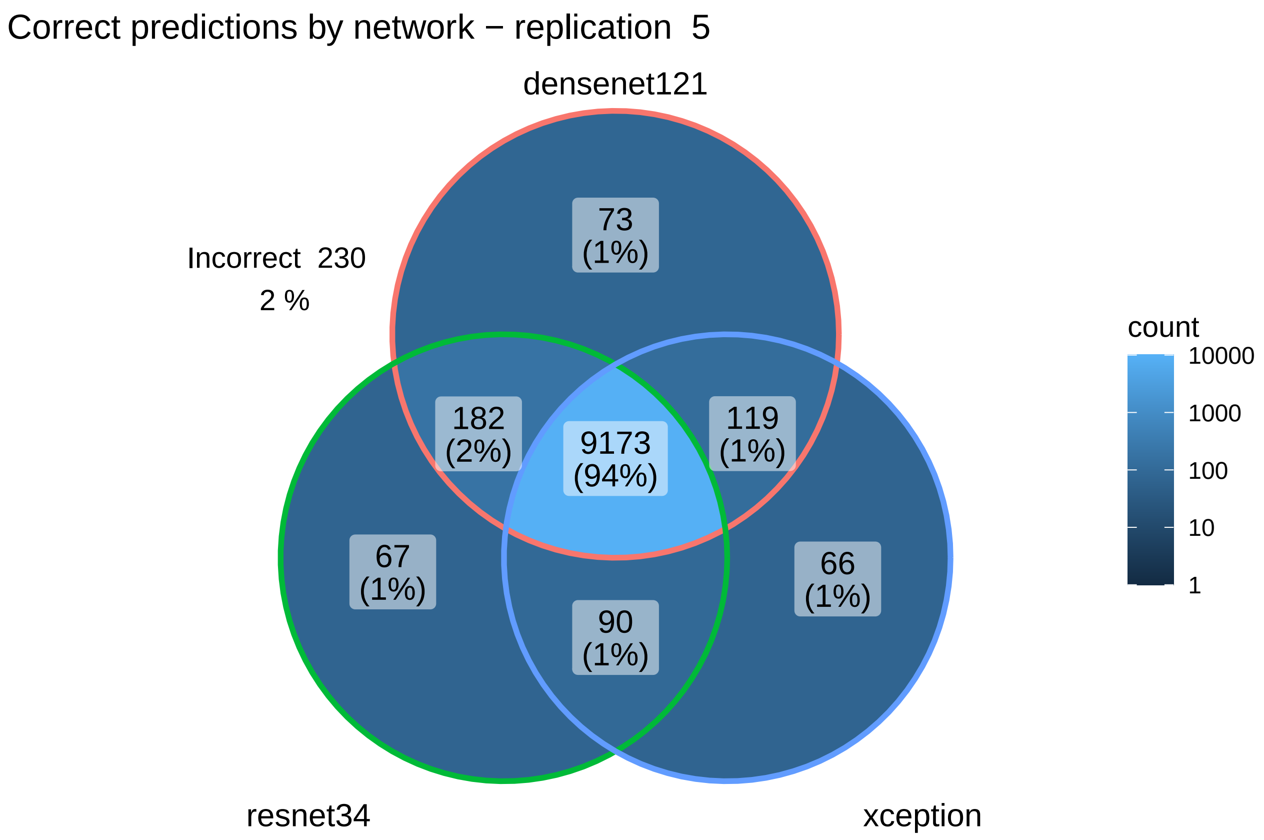
```
for (ri in 1:r_n)
{
  nets_cor_list <- list()
  incor <- 1:samples_n
  for (ni in 1:nets_n)
  {
    cor_list <- which(nets_test_cor_preds[ri, ni, ])
    nets_cor_list[[nets_outputs$networks[ni]]] = cor_list
    incor <- setdiff(incor, cor_list)
  }
  incor_n <- length(incor)
  venn_diag <- ggVennDiagram(nets_cor_list) + scale_fill_gradient(trans="log10", name="count", limits=c
    annotate(geom="text", x=-4, y=5, label=paste("Incorrect ", incor_n, "\n", round(incor_n / samples_n
    ggtitle(paste("Correct predictions by network - replication ", ri)) +
    scale_x_continuous(limits=c(-8, 10))
  print(venn_diag)
}
```

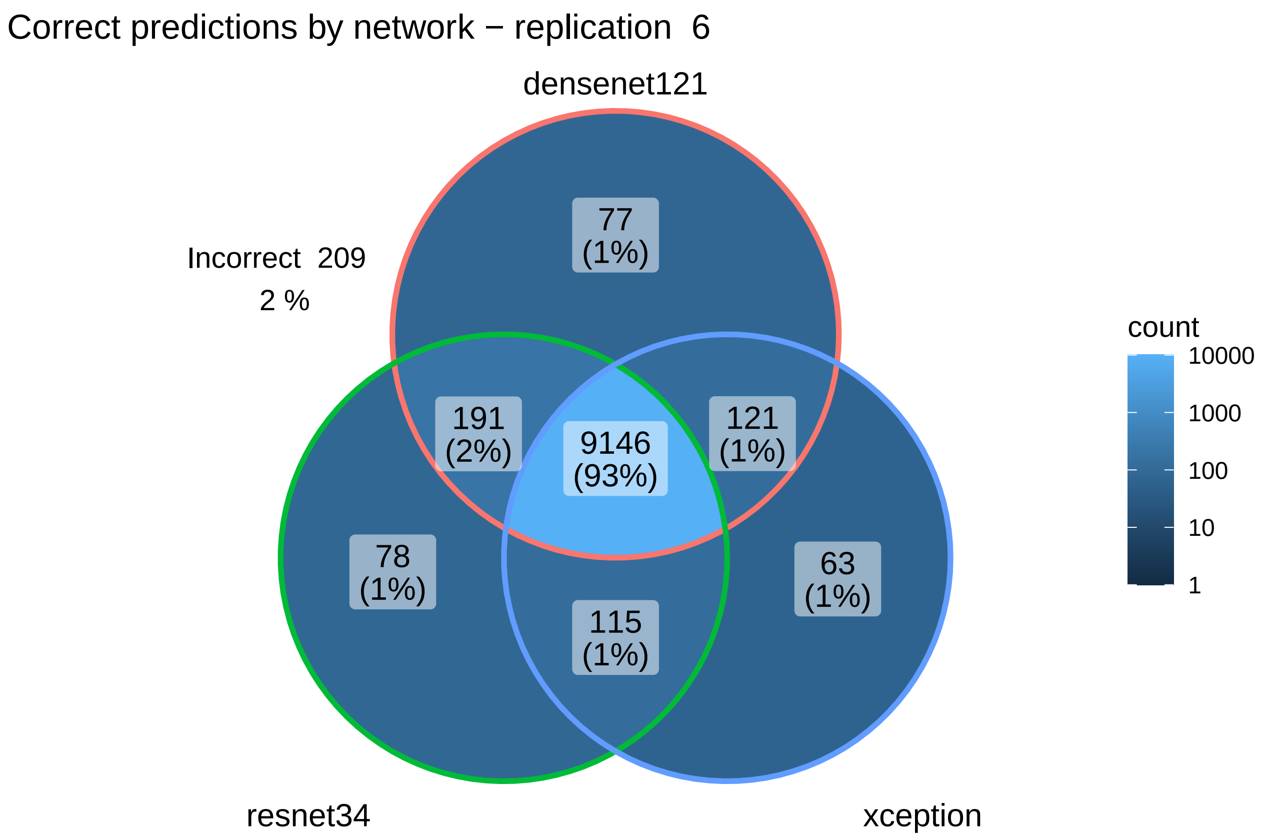

Correct predictions by network − replication 1
densenet121
Incorrect 212
2 %
83
(1%)
178
(2%)
9160
(94%)
119
(1%)
70
(1%)
59
(1%)
119
(1%)
resnet34
xception
count
10000
1000
100
10
1

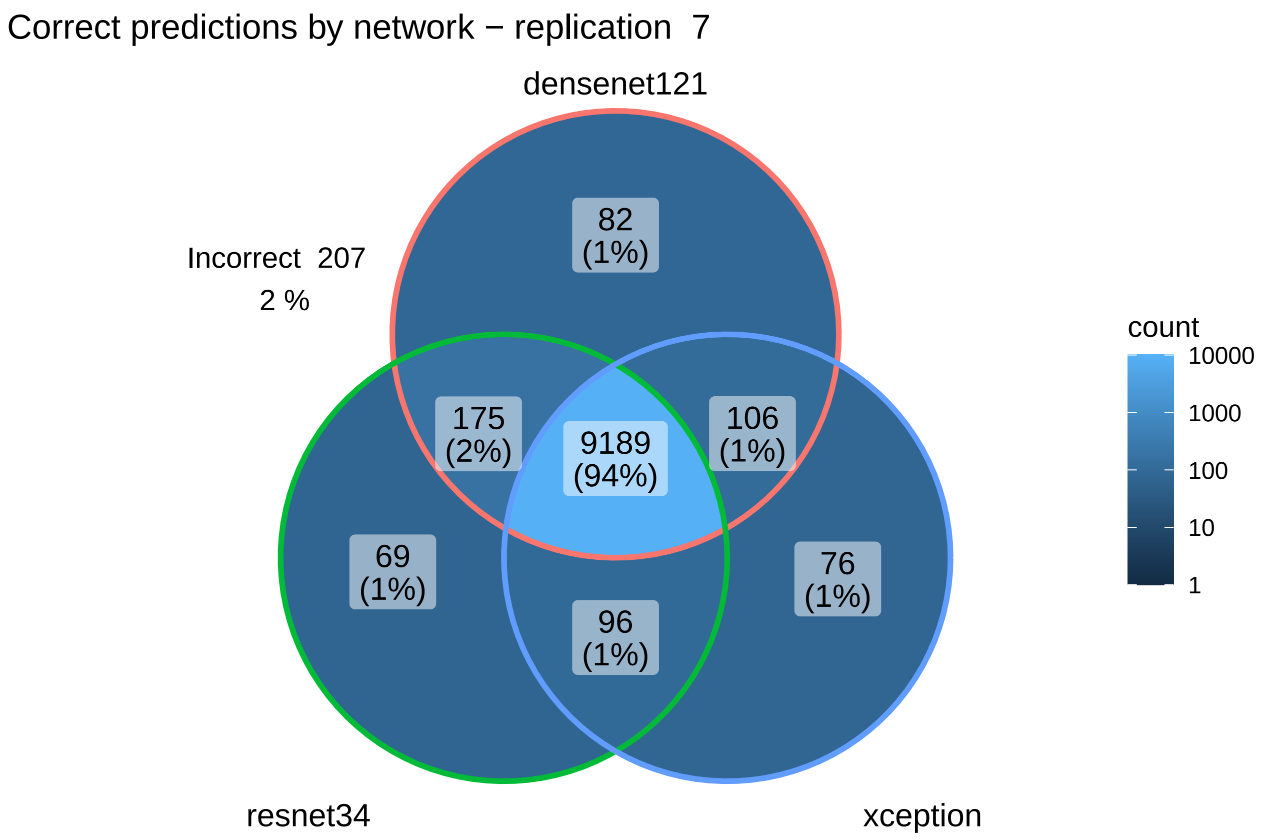

Correct predictions by network − replication 2
densenet121
Incorrect 204
2 %
79
(1%)
153
(2%)
9206
(94%)
109
(1%)
84
(1%)
105
(1%)
60
(1%)
resnet34
xception
count
10000
1000
100
10
1

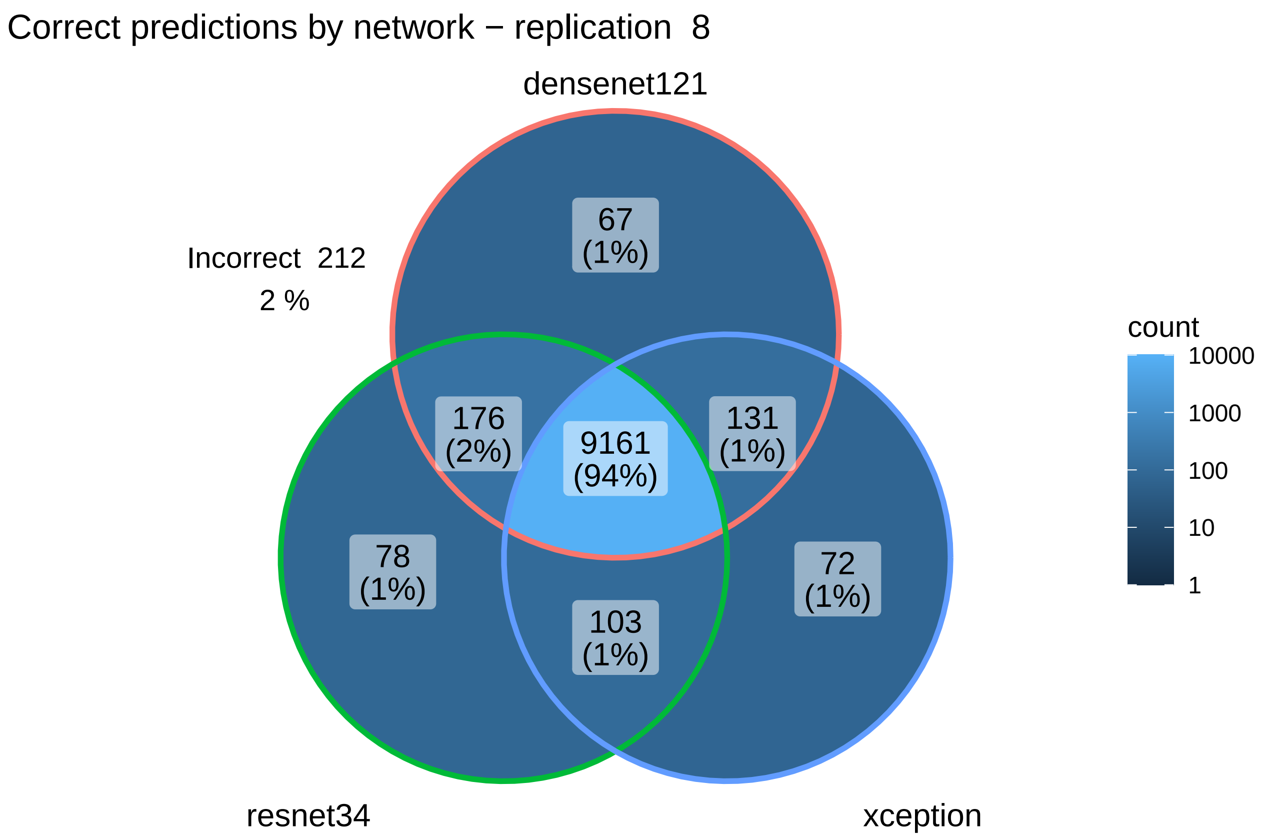

Correct predictions by network − replication 3
densenet121
resnet34
xception
Incorrect 204
2 %
78
(1%)
178
(2%)
9157
(93%)
120
(1%)
83
(1%)
113
(1%)
67
(1%)
count
10000
1000
100
10
1

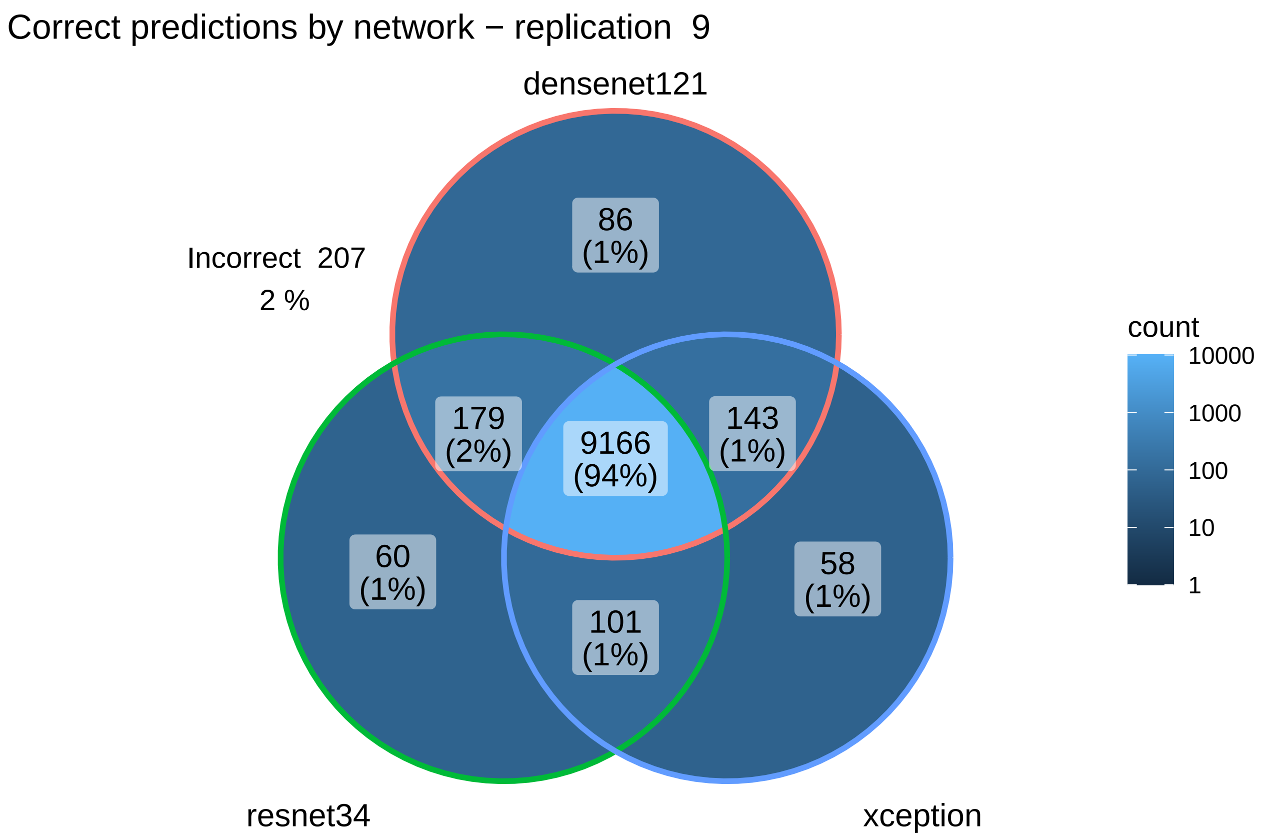

Correct predictions by network − replication 4
densenet121
resnet34
xception
Incorrect 192
2 %
93
(1%)
162
(2%)
9171
(94%)
130
(1%)
92
(1%)
109
(1%)
51
(1%)
count
10000
1000
100
10
1

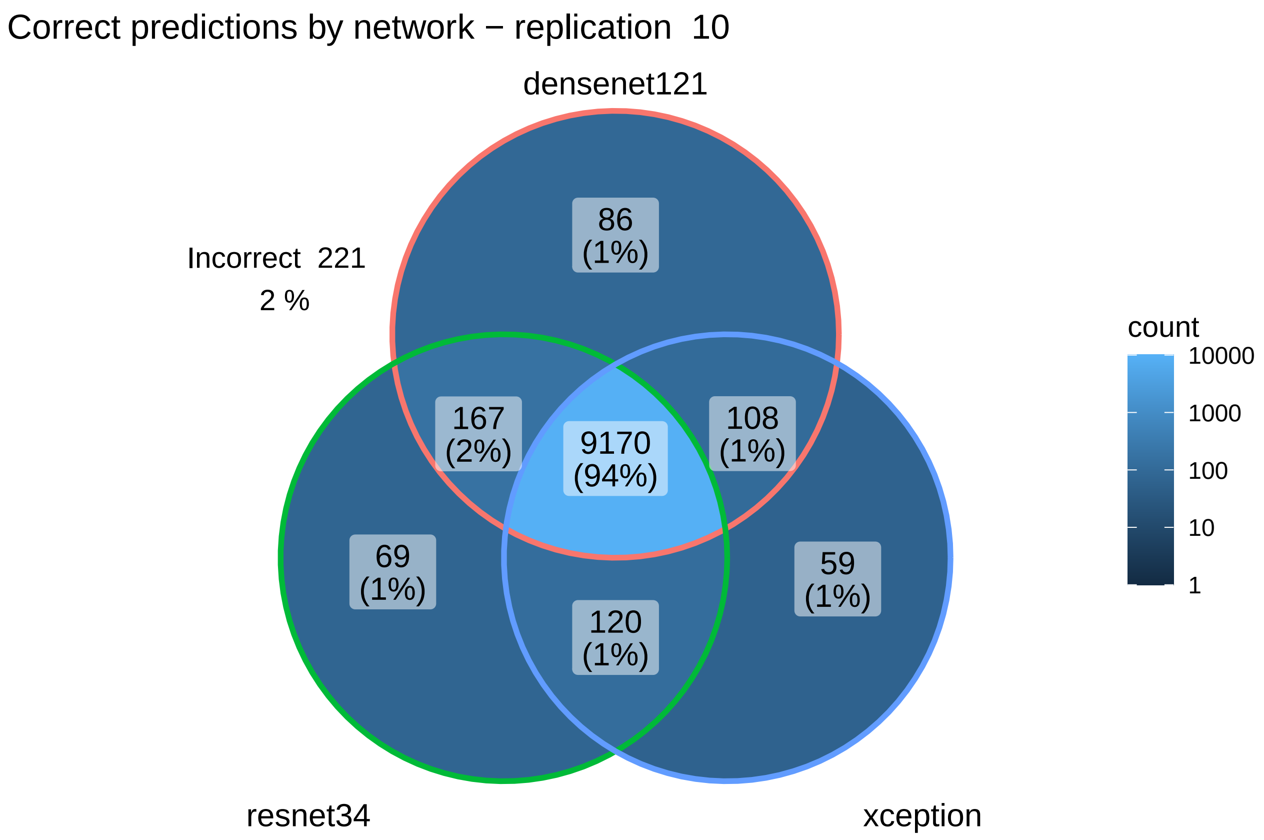

Correct predictions by network − replication 5
densenet121
resnet34
xception
Incorrect 230
2 %
73
(1%)
182
(2%)
9173
(94%)
119
(1%)
67
(1%)
90
(1%)
66
(1%)
count
10000
1000
100
10
1

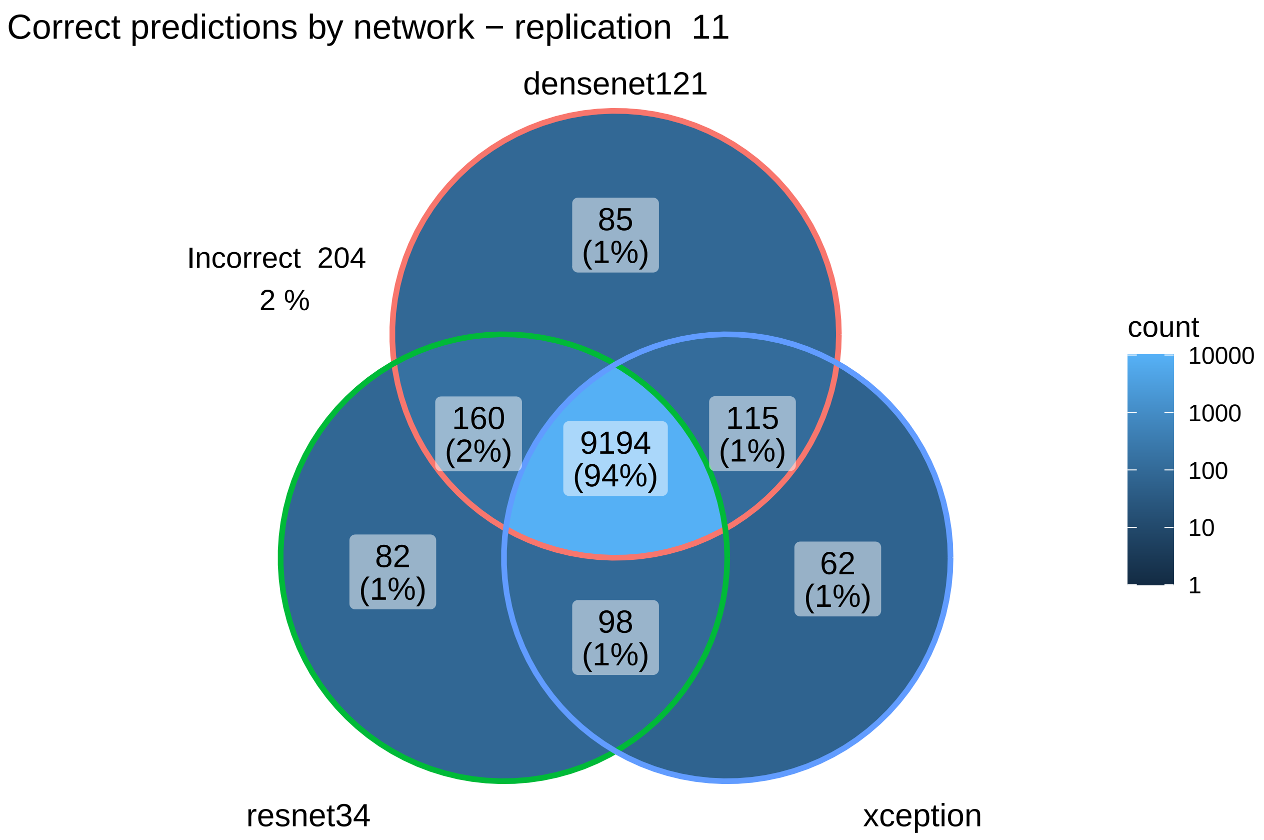

Correct predictions by network − replication 6
densenet121
resnet34
xception
Incorrect 209
2 %
77 (1%)
191 (2%)
121 (1%)
9146 (93%)
78 (1%)
115 (1%)
63 (1%)
count
10000
1000
100
10
1

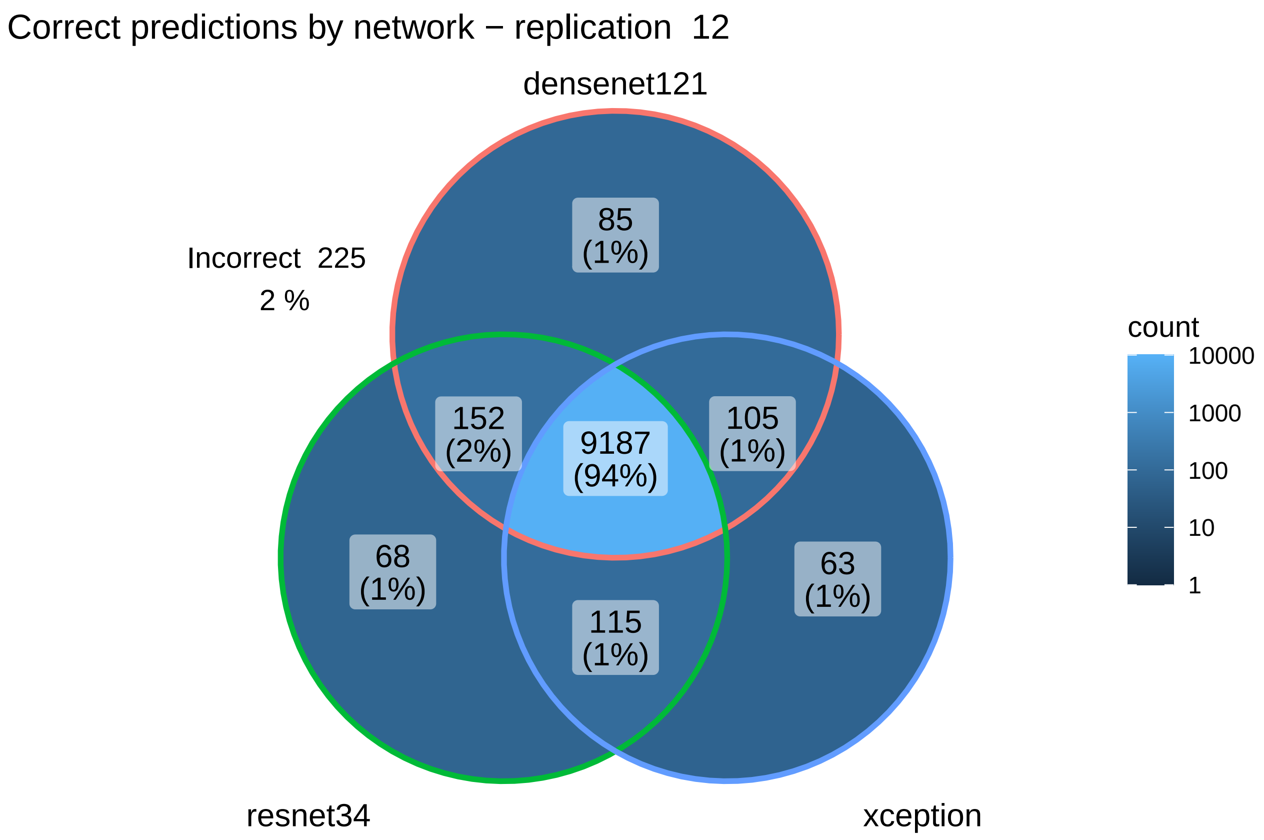

Correct predictions by network − replication 7
densenet121
Incorrect 207
2 %
82
(1%)
175
(2%)
9189
(94%)
106
(1%)
69
(1%)
96
(1%)
76
(1%)
resnet34
xception
count
10000
1000
100
10
1

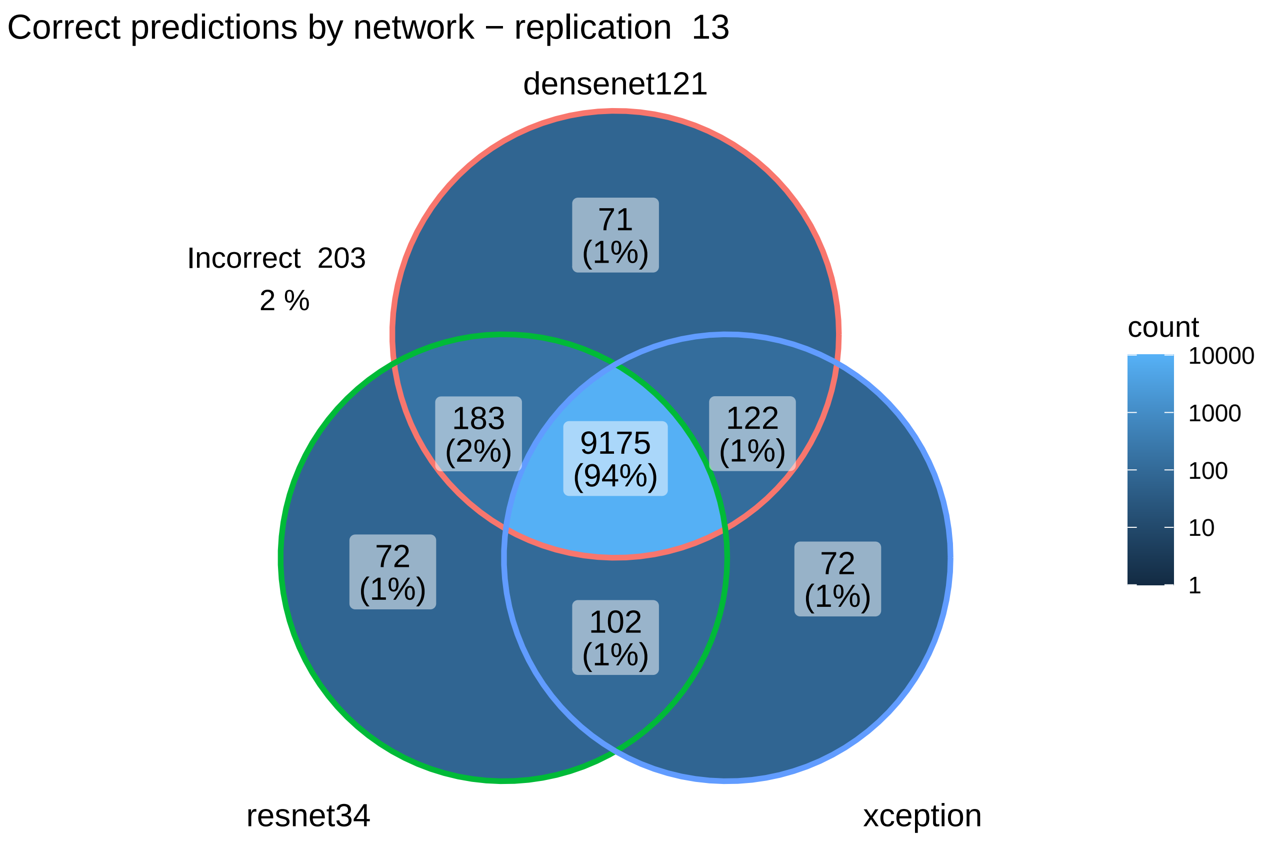

Correct predictions by network − replication 8
densenet121
resnet34
xception
Incorrect 212
2 %
67
(1%)
176
(2%)
9161
(94%)
131
(1%)
78
(1%)
103
(1%)
72
(1%)
count
10000
1000
100
10
1

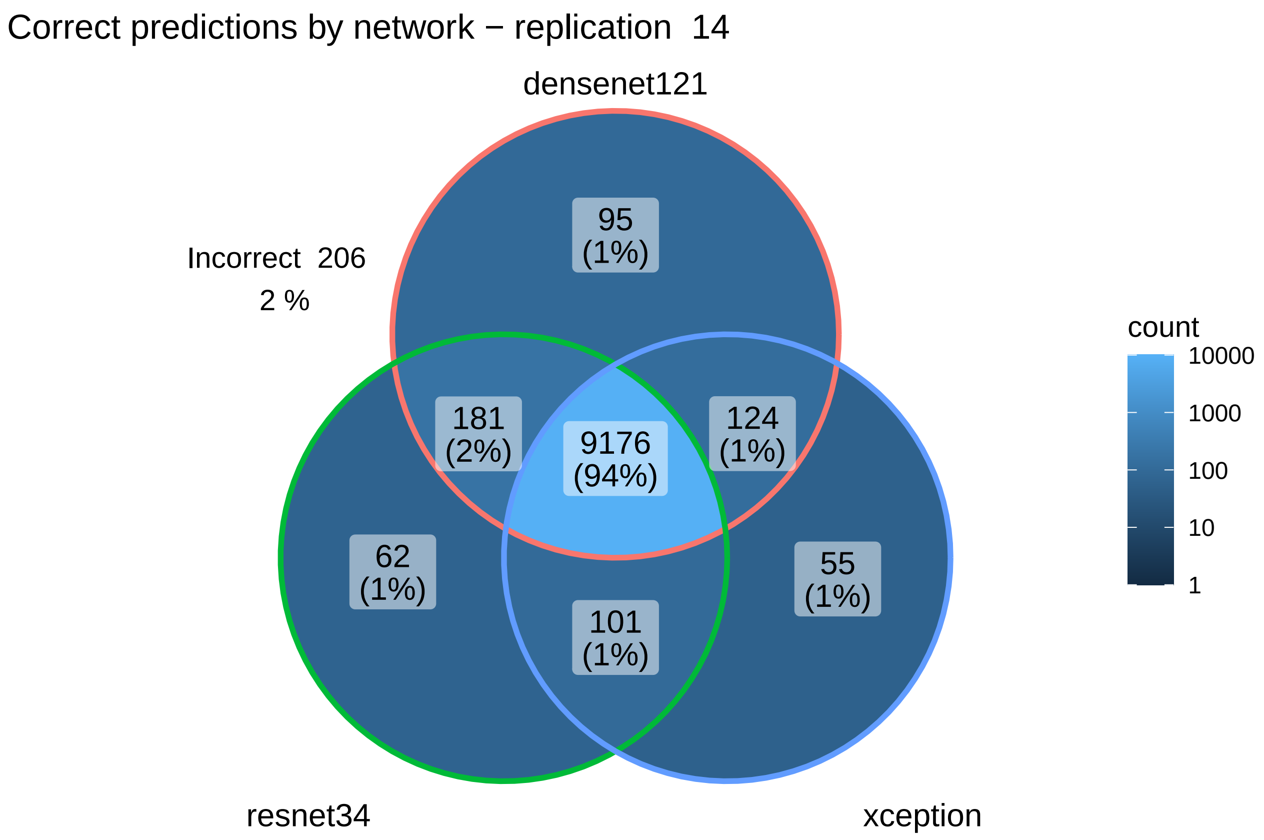

Correct predictions by network − replication 9
densenet121
Incorrect 207
2 %
86
(1%)
179
(2%)
9166
(94%)
143
(1%)
60
(1%)
101
(1%)
58
(1%)
resnet34
xception
count
10000
1000
100
10
1

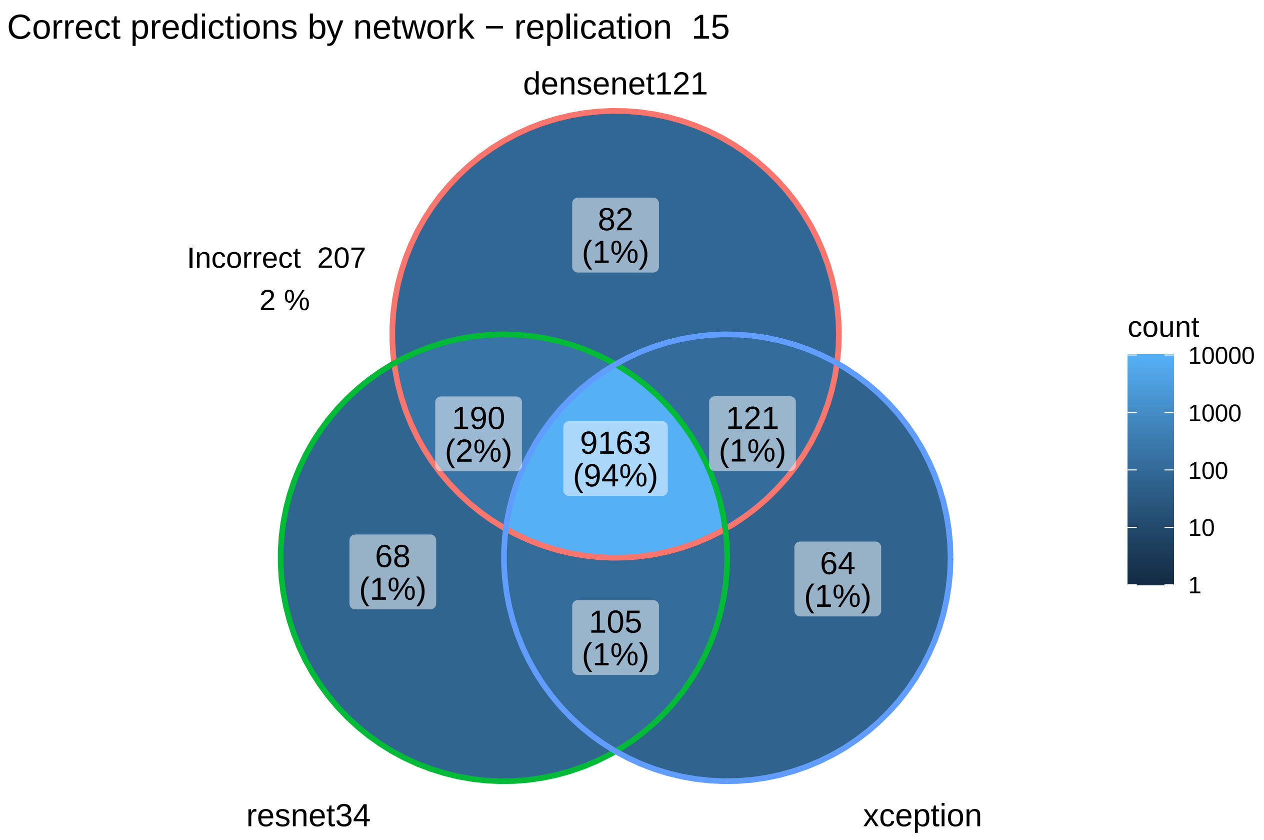

Correct predictions by network − replication 10
densenet121
Incorrect 221
2 %
86
(1%)
167
(2%)
9170
(94%)
108
(1%)
69
(1%)
120
(1%)
59
(1%)
resnet34
xception
count
10000
1000
100
10
1

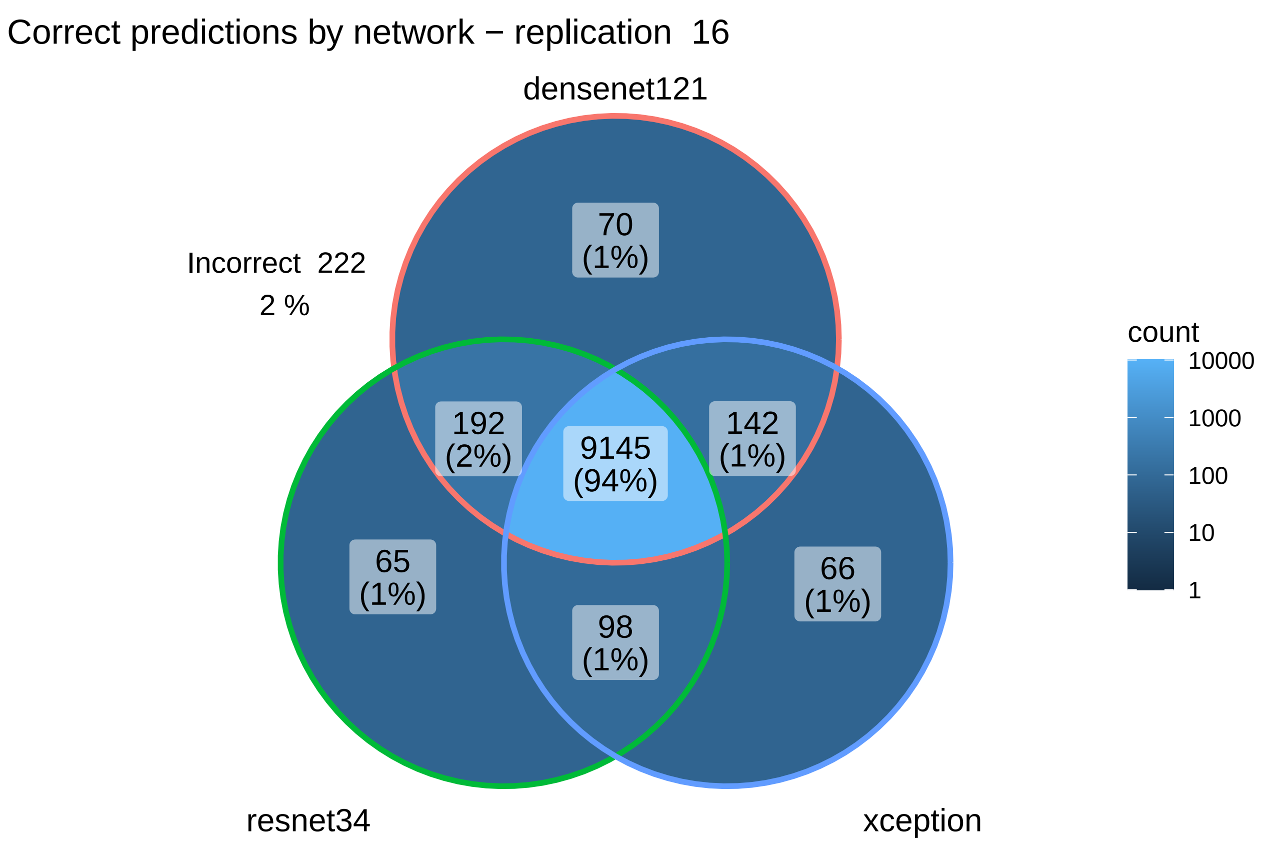

Correct predictions by network − replication 11
densenet121
Incorrect 204
2 %
85
(1%)
160
(2%)
9194
(94%)
115
(1%)
82
(1%)
98
(1%)
62
(1%)
resnet34
xception
count
10000
1000
100
10
1


Correct predictions by network − replication 12
densenet121
Incorrect 225
2 %
85
(1%)
152
(2%)
9187
(94%)
105
(1%)
68
(1%)
115
(1%)
63
(1%)
resnet34
xception
count
10000
1000
100
10
1

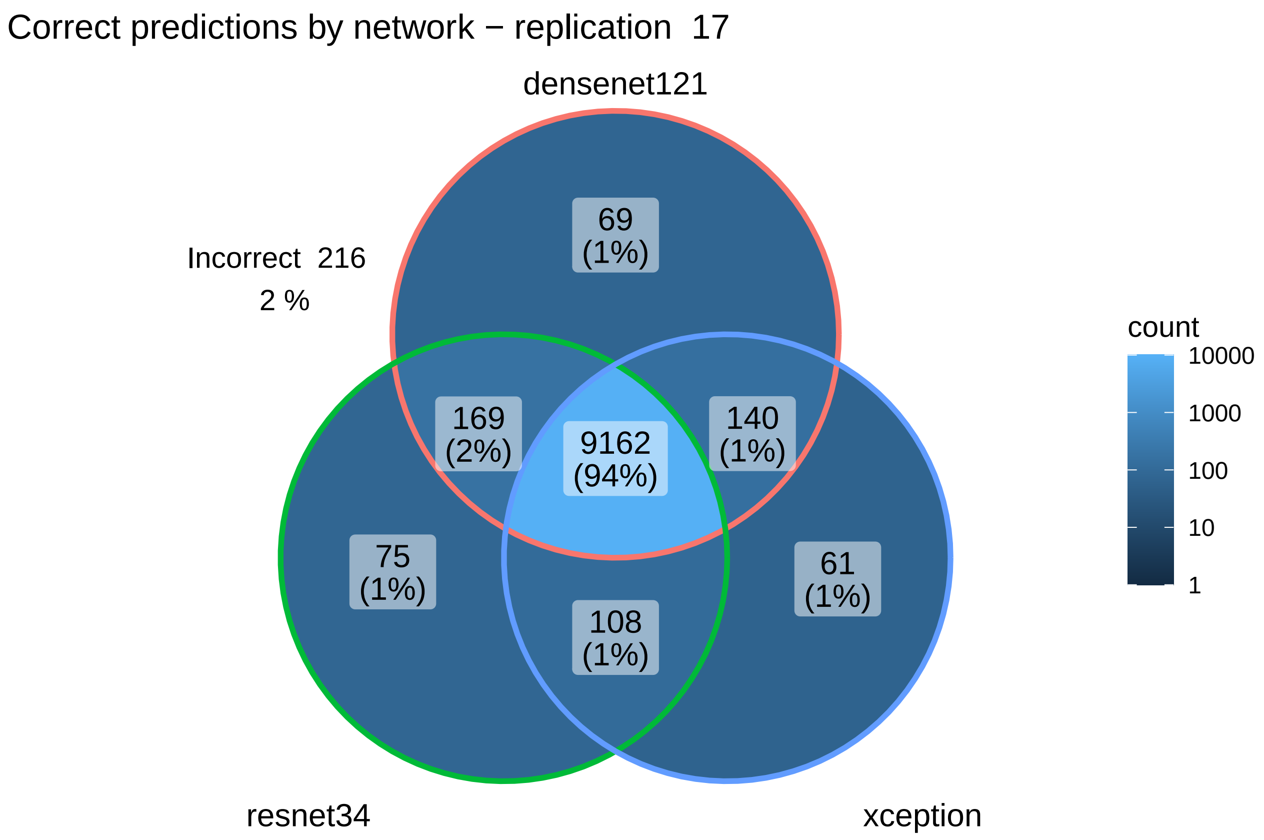

Correct predictions by network − replication 13
densenet121
Incorrect 203
2 %
71
(1%)
183
(2%)
9175
(94%)
122
(1%)
72
(1%)
102
(1%)
72
(1%)
resnet34
xception
count
10000
1000
100
10
1


Correct predictions by network − replication 14
densenet121
Incorrect 206
2 %
95 (1%)
181 (2%)
9176 (94%)
124 (1%)
62 (1%)
101 (1%)
55 (1%)
resnet34
xception
count
10000
1000
100
10
1

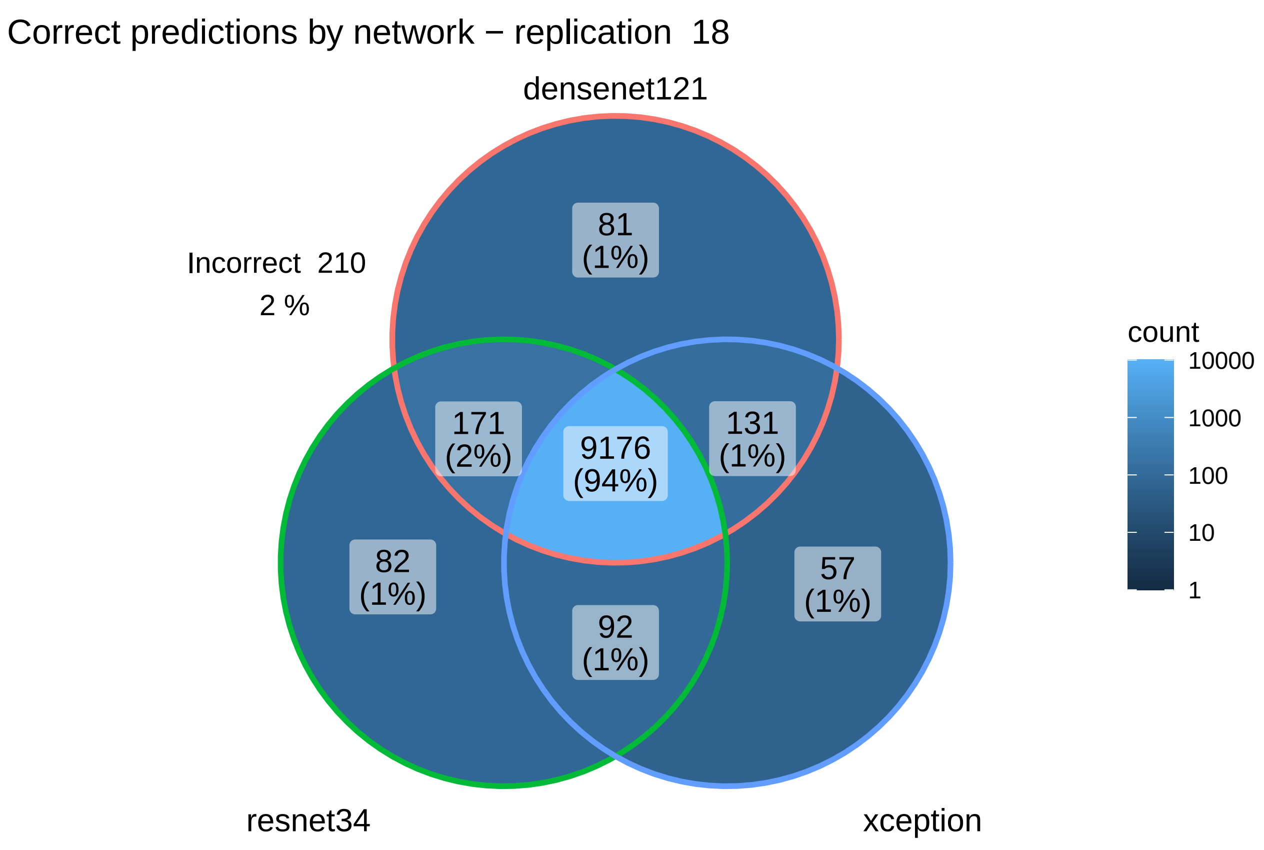

Correct predictions by network − replication 15
densenet121
Incorrect 207
2 %
82
(1%)
190
(2%)
9163
(94%)
121
(1%)
68
(1%)
105
(1%)
64
(1%)
resnet34
xception
count
10000
1000
100
10
1

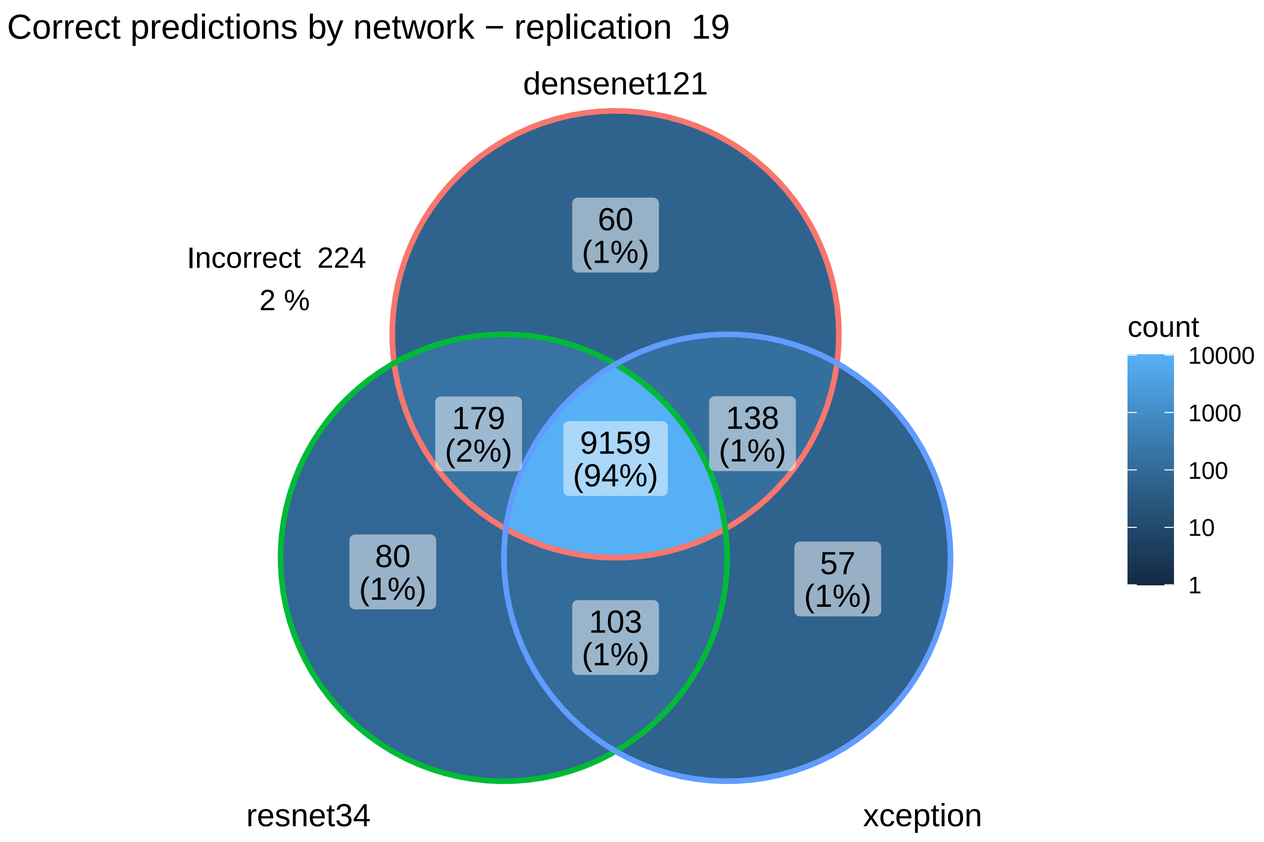


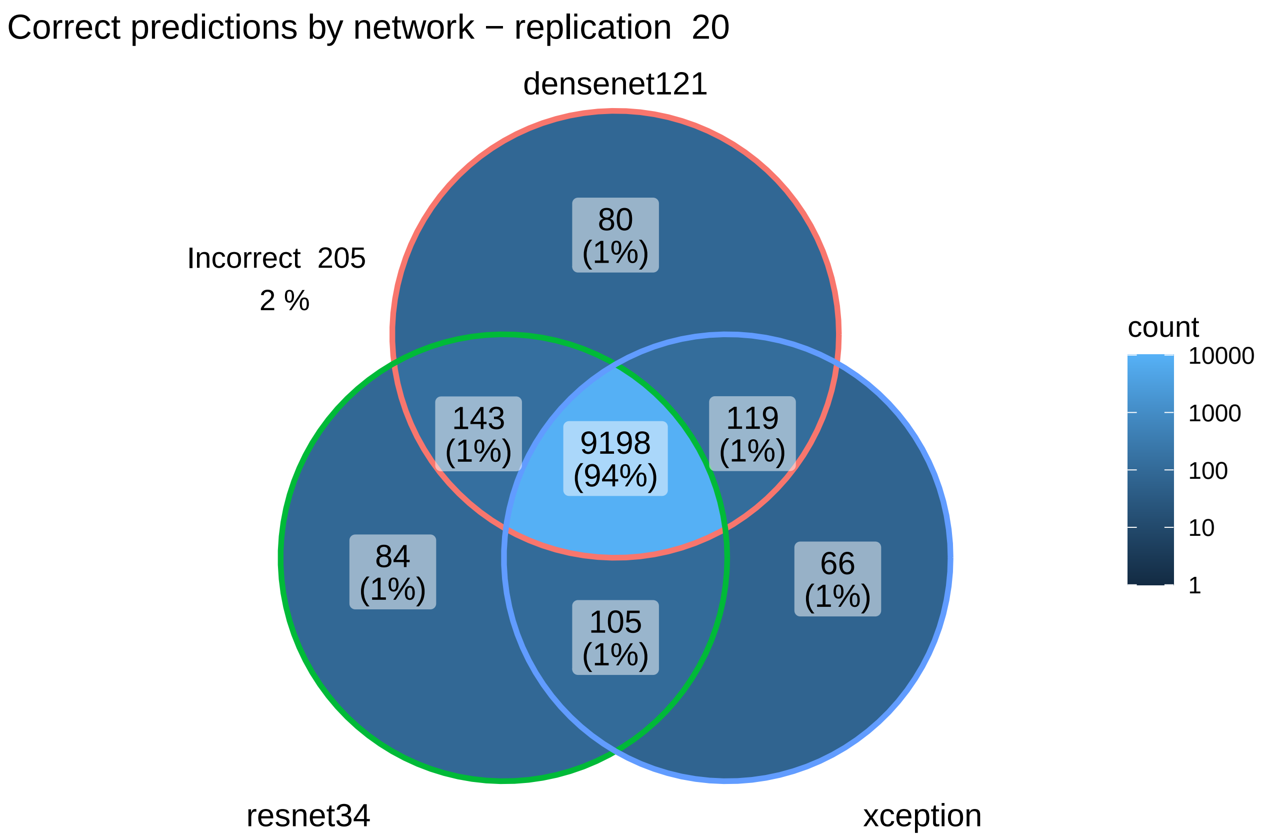
Correct predictions by network − replication 16
densenet121
Incorrect 222
2 %
70
(1%)
192
(2%)
9145
(94%)
142
(1%)
65
(1%)
98
(1%)
66
(1%)
resnet34
xception
count
10000
1000
100
10
1

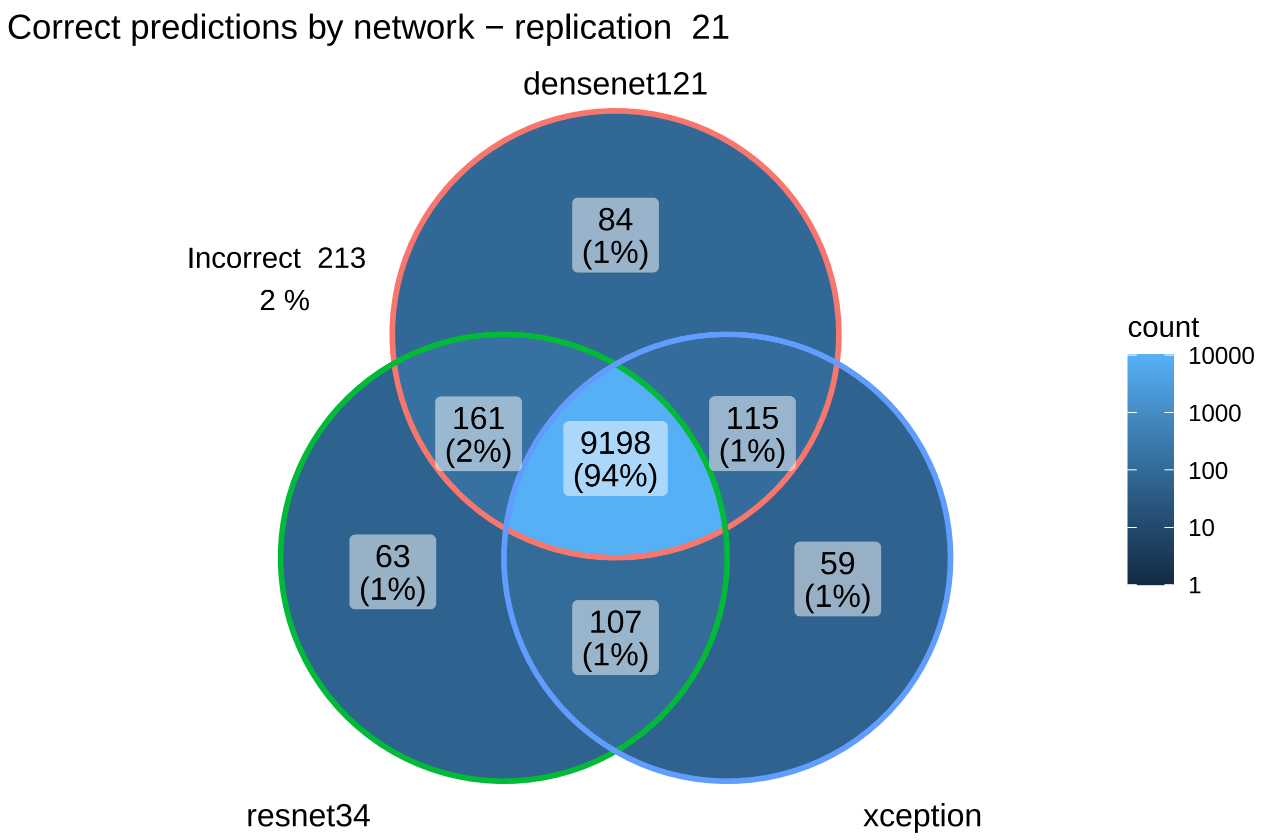


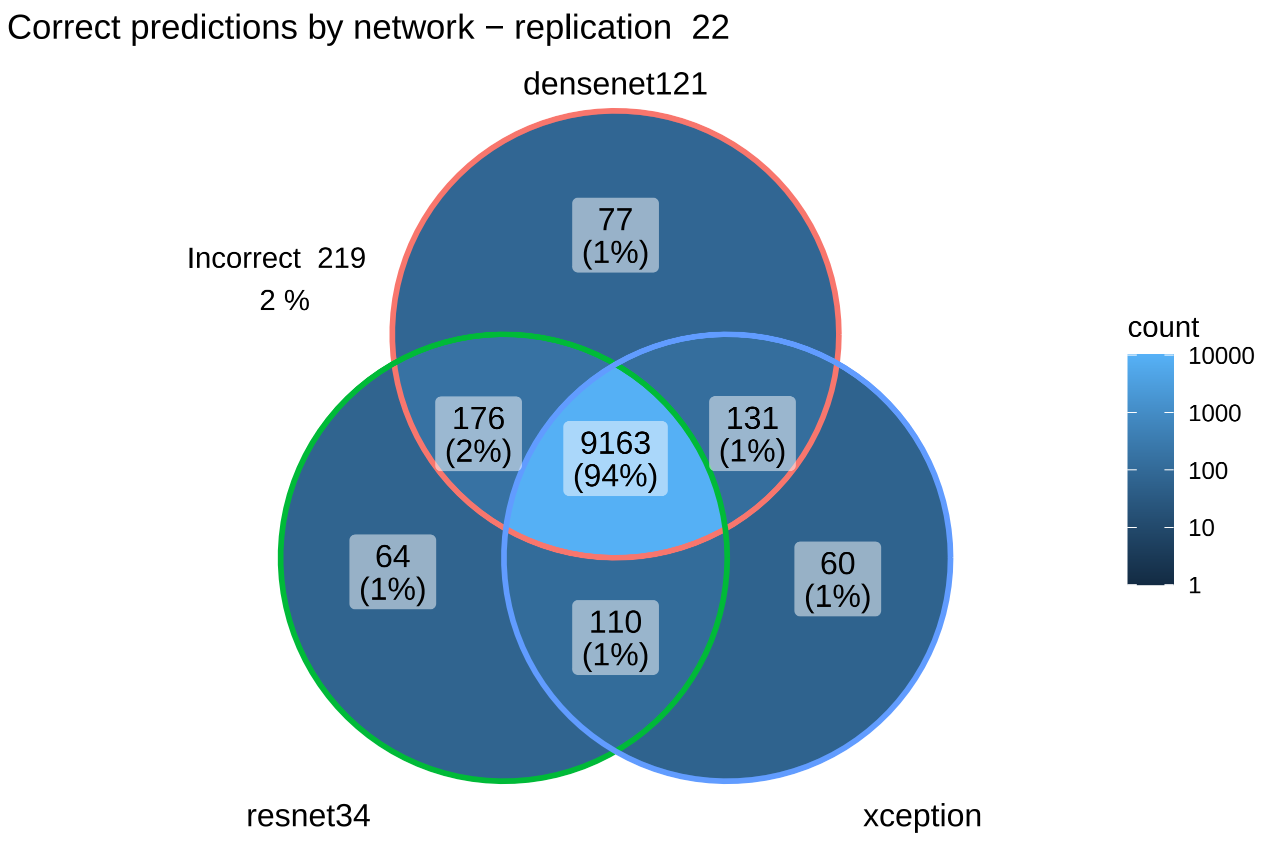
Correct predictions by network − replication 17
densenet121
resnet34
xception
Incorrect 216
2 %
69 (1%)
169 (2%)
9162 (94%)
140 (1%)
75 (1%)
108 (1%)
61 (1%)
count
10000
1000
100
10
1

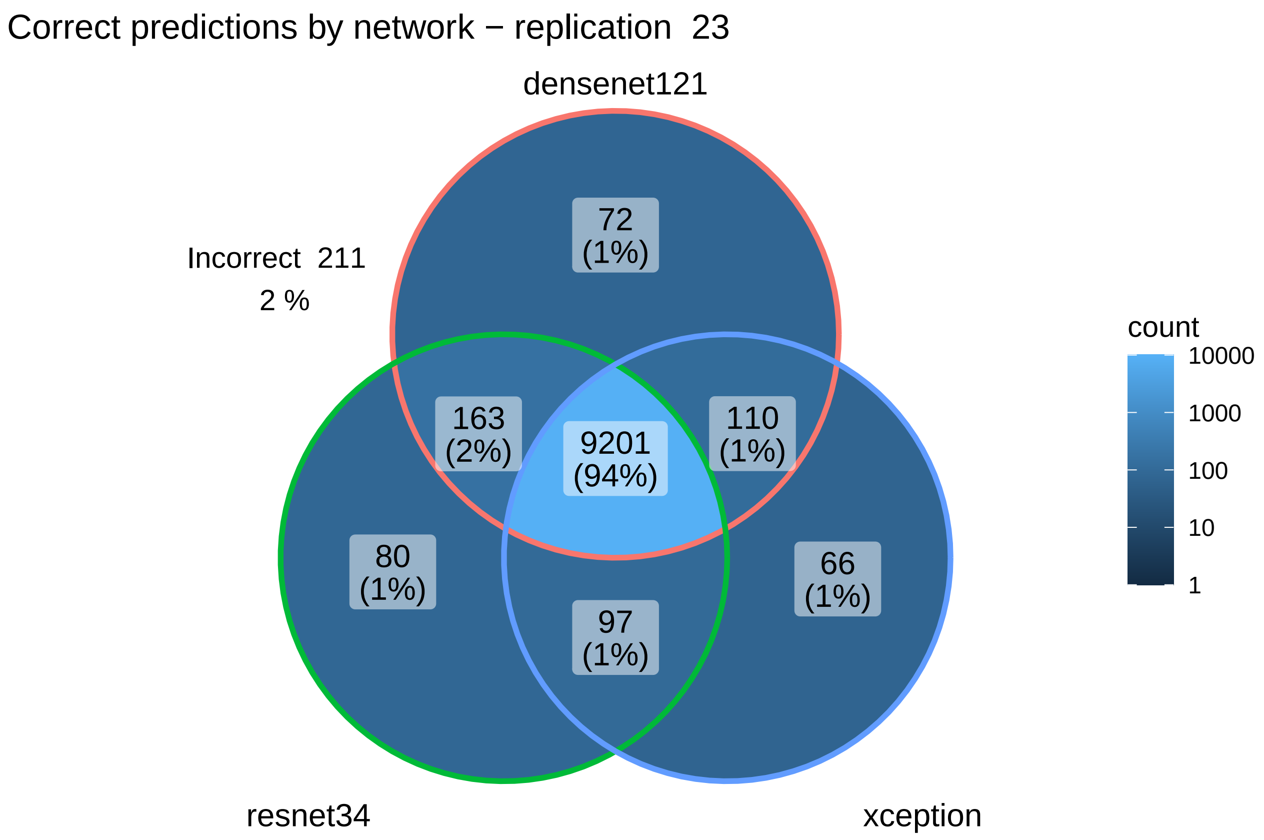

Correct predictions by network − replication 18
densenet121
Incorrect 210
2 %
81
(1%)
171
(2%)
9176
(94%)
131
(1%)
82
(1%)
92
(1%)
57
(1%)
resnet34
xception
count
10000
1000
100
10
1

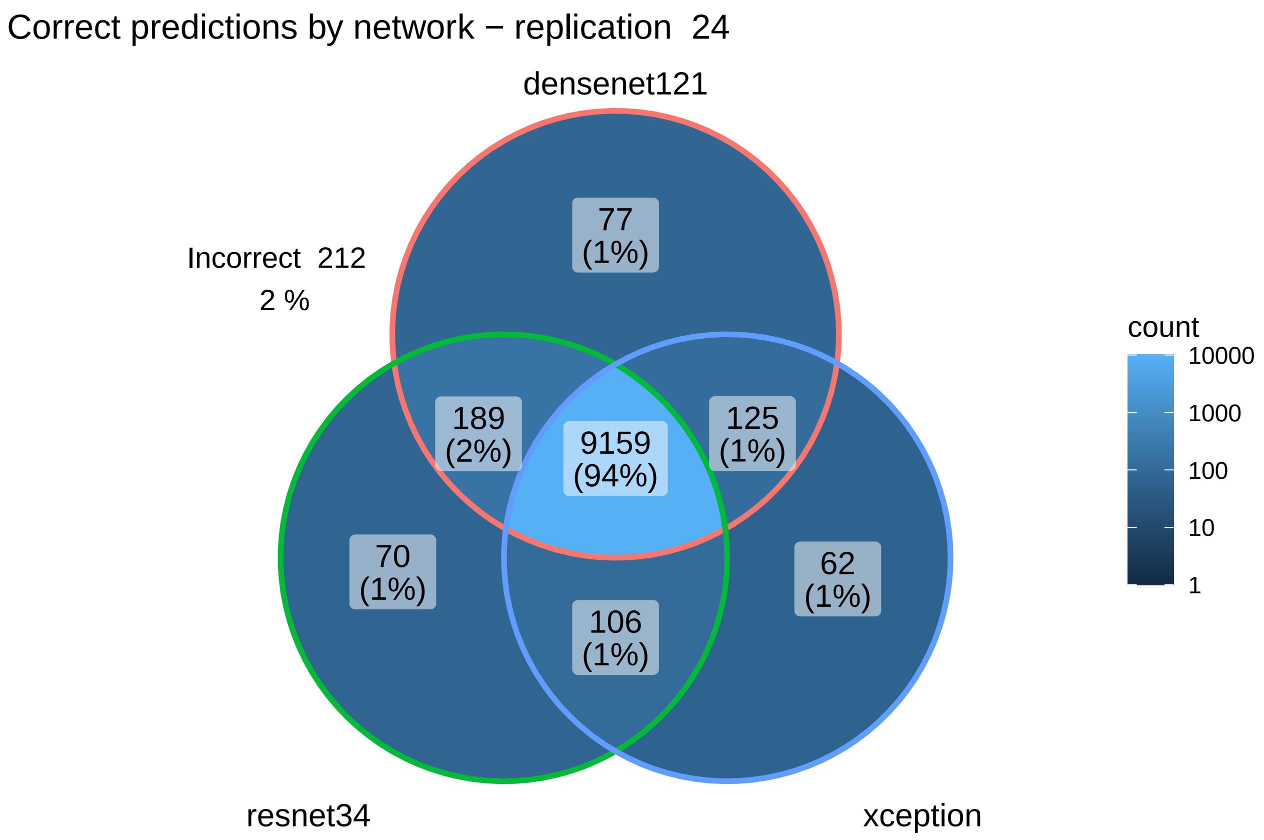

Correct predictions by network − replication 19
densenet121
Incorrect 224
2 %
60
(1%)
179
(2%)
9159
(94%)
138
(1%)
80
(1%)
103
(1%)
57
(1%)
resnet34
xception
count
10000
1000
100
10
1


Correct predictions by network − replication 20
densenet121
Incorrect 205
2 %
80
(1%)
143
(1%)
9198
(94%)
119
(1%)
84
(1%)
105
(1%)
66
(1%)
resnet34
xception
count
10000
1000
100
10
1

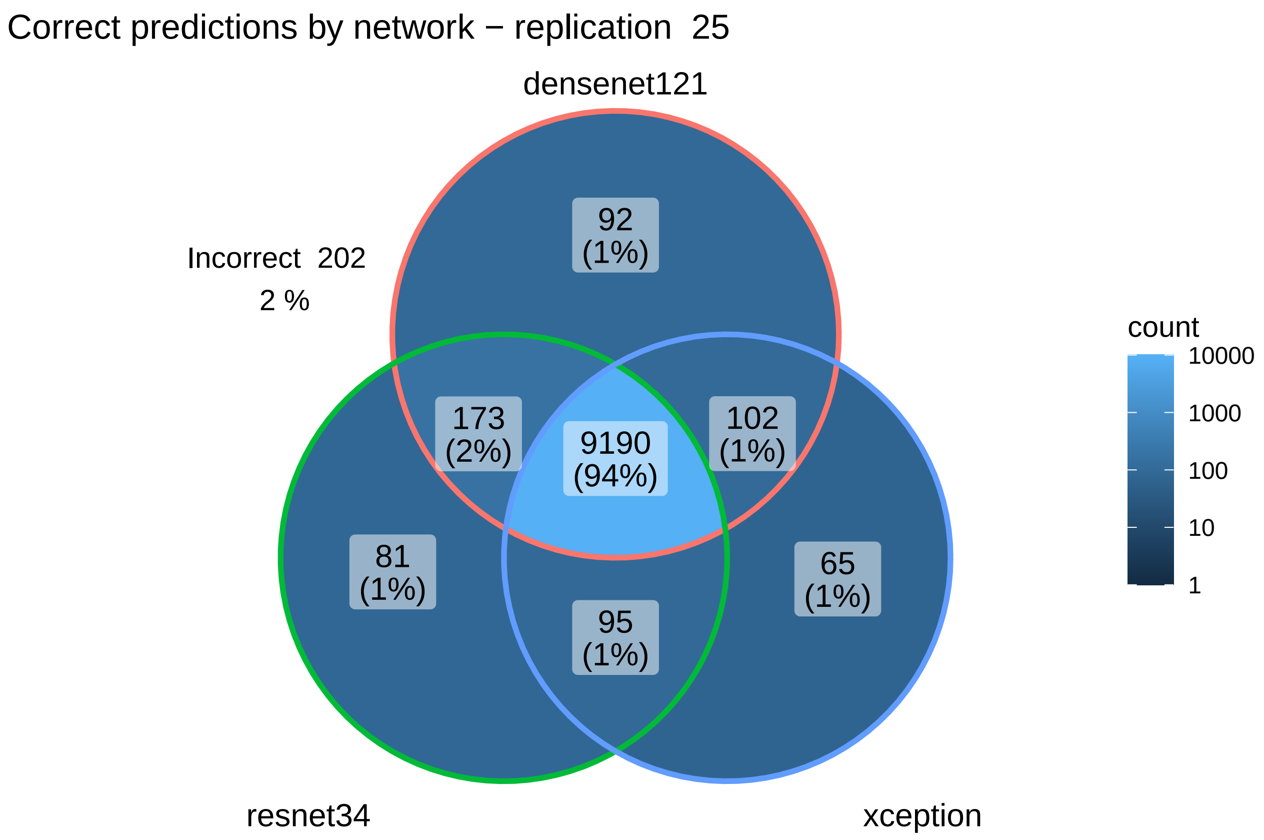

Correct predictions by network − replication 21
densenet121
Incorrect 213
2 %
84
(1%)
161
(2%)
9198
(94%)
115
(1%)
63
(1%)
107
(1%)
59
(1%)
resnet34
xception
count
10000
1000
100
10
1

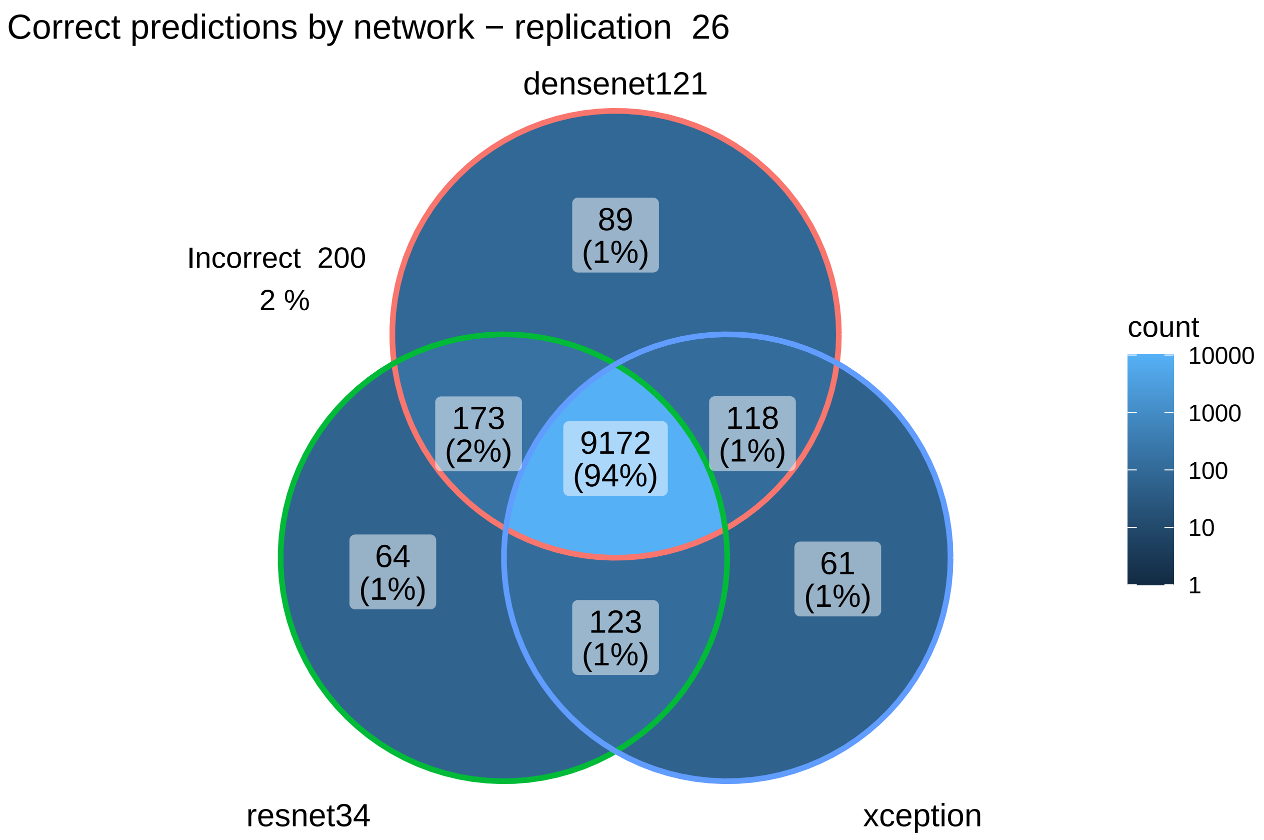

Correct predictions by network − replication 22
densenet121
Incorrect 219
2 %
77
(1%)
176
(2%)
9163
(94%)
131
(1%)
64
(1%)
110
(1%)
60
(1%)
resnet34
xception
count
10000
1000
100
10
1

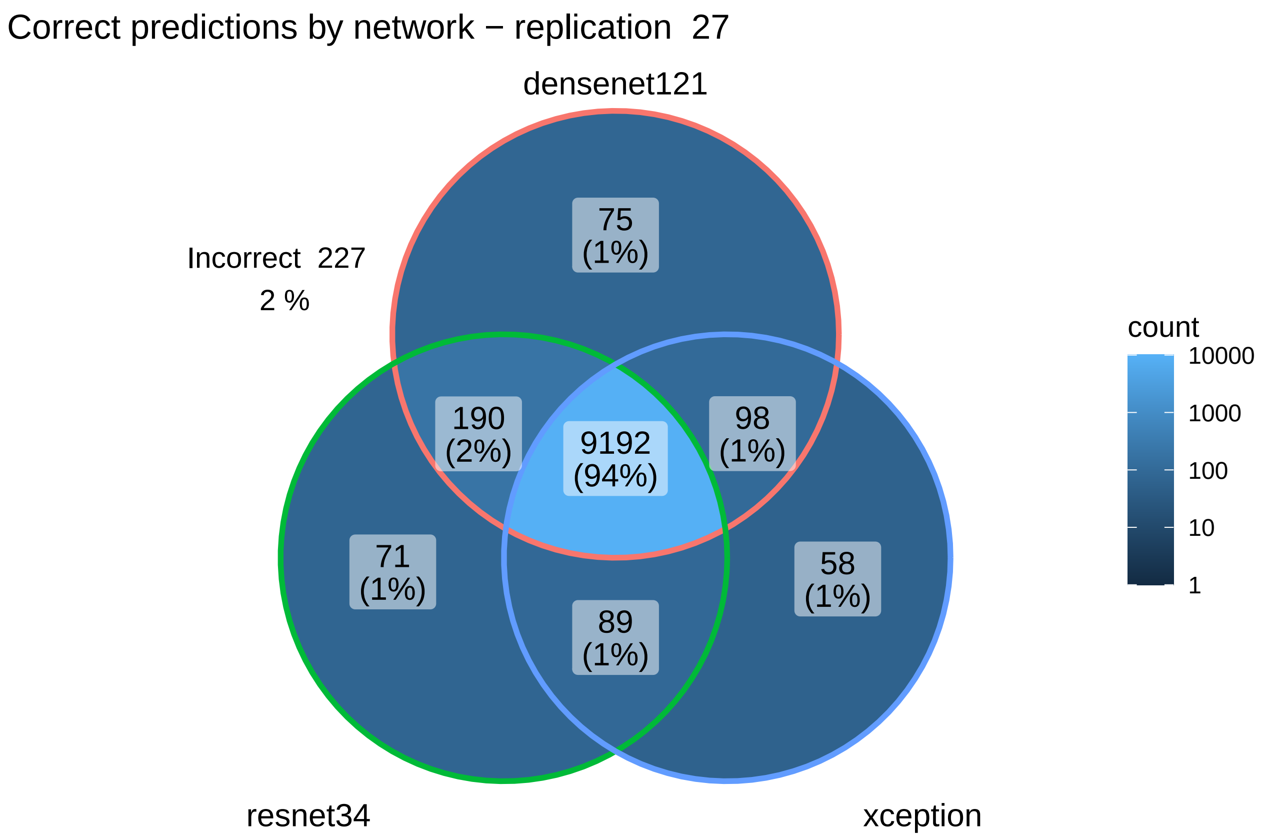

Correct predictions by network − replication 23
densenet121
Incorrect 211
2 %
72
(1%)
163
(2%)
9201
(94%)
110
(1%)
80
(1%)
97
(1%)
66
(1%)
resnet34
xception
count
10000
1000
100
10
1

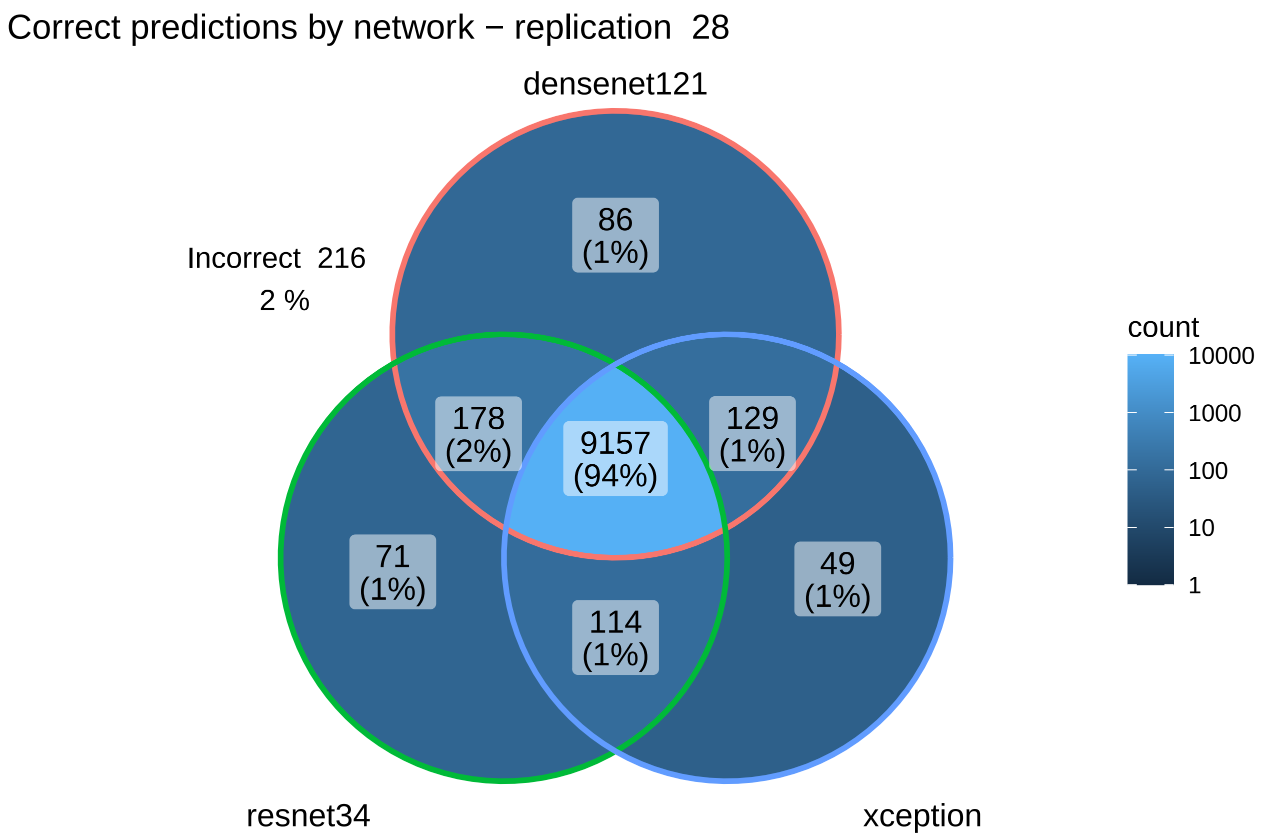


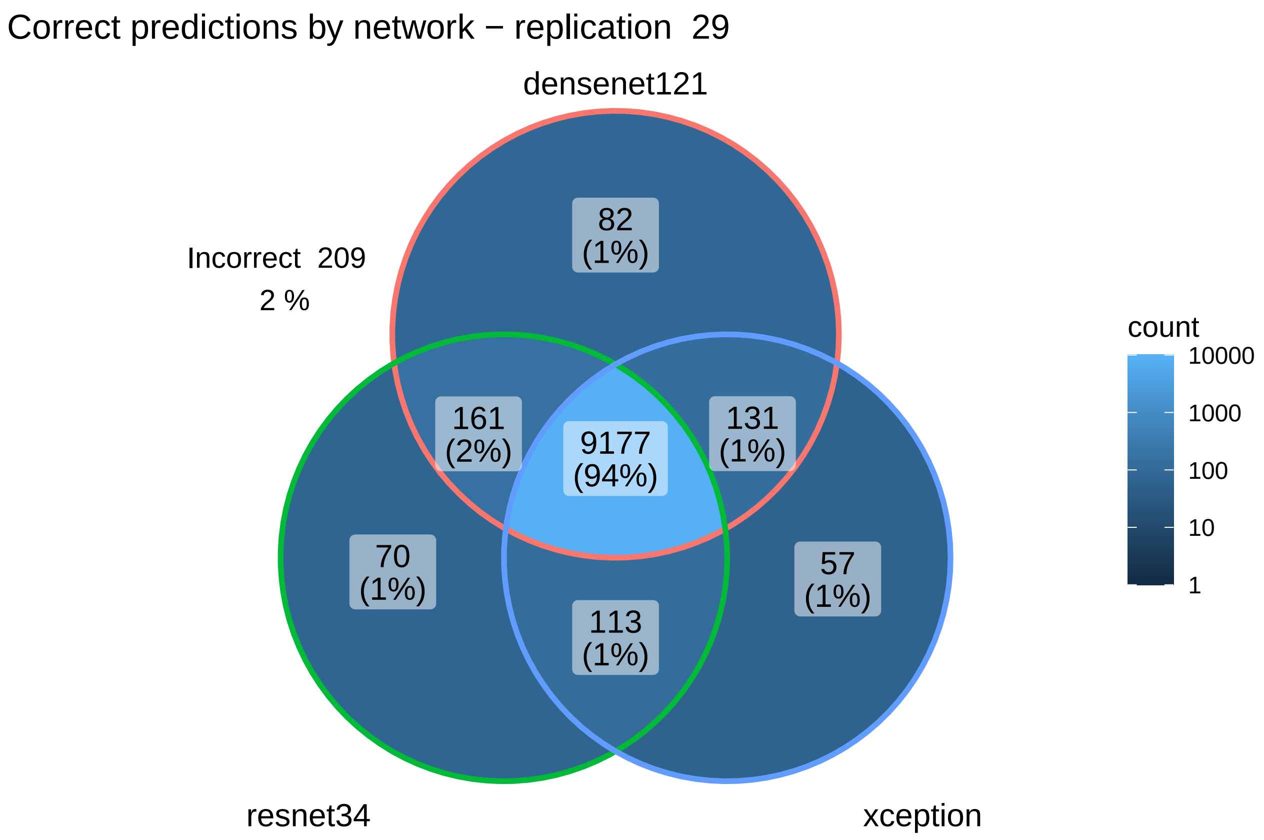
Correct predictions by network − replication 24
densenet121
Incorrect 212
2 %
77
(1%)
189
(2%)
9159
(94%)
125
(1%)
70
(1%)
106
(1%)
62
(1%)
resnet34
xception
count
10000
1000
100
10
1

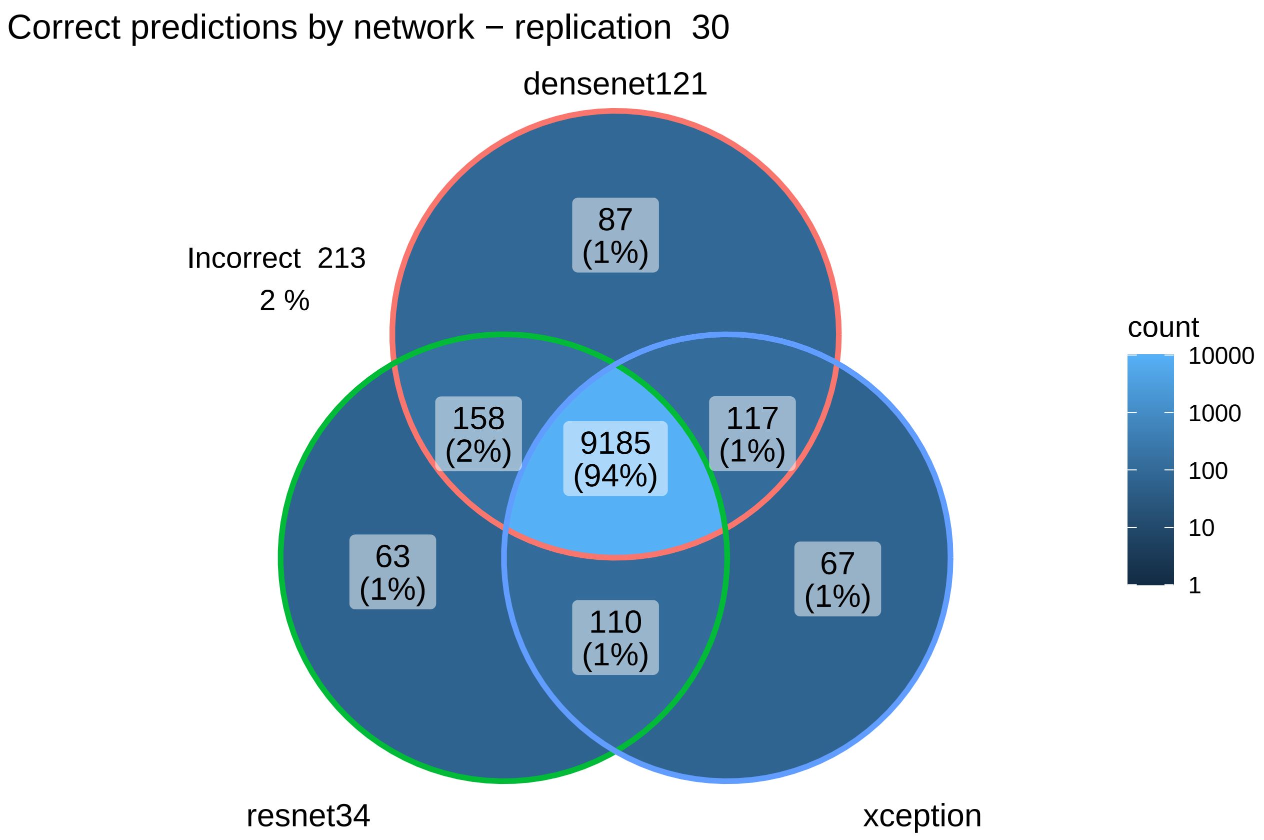

Correct predictions by network − replication 25
densenet121
Incorrect 202
2 %
92
(1%)
173
(2%)
9190
(94%)
102
(1%)
81
(1%)
95
(1%)
65
(1%)
resnet34
xception
count
10000
1000
100
10
1


Correct predictions by network − replication 26
densenet121
Incorrect 200
2 %
89
(1%)
173
(2%)
9172
(94%)
118
(1%)
64
(1%)
123
(1%)
61
(1%)
resnet34
xception
count
10000
1000
100
10
1


Correct predictions by network − replication 27
densenet121
resnet34
xception
Incorrect 227
2 %
75
(1%)
190
(2%)
9192
(94%)
98
(1%)
71
(1%)
89
(1%)
58
(1%)
count
10000
1000
100
10
1


Correct predictions by network − replication 28
densenet121
Incorrect 216
2 %
86
(1%)
178
(2%)
9157
(94%)
129
(1%)
71
(1%)
114
(1%)
49
(1%)
resnet34
xception
count
10000
1000
100
10
1


Correct predictions by network − replication 29
densenet121
Incorrect 209
2 %
82
(1%)
161
(2%)
9177
(94%)
131
(1%)
70
(1%)
113
(1%)
57
(1%)
resnet34
xception
count
10000
1000
100
10
1

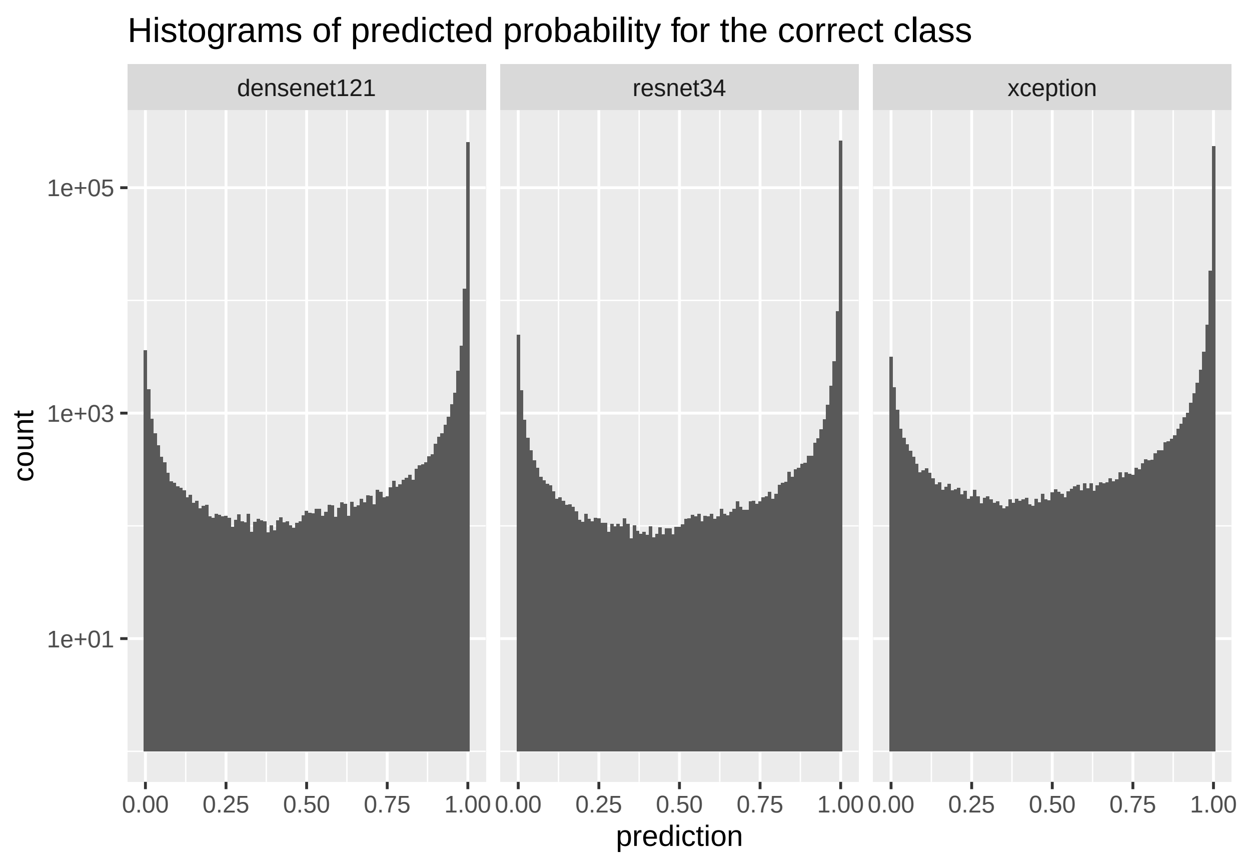


In all replications, around 94% of data was correctly classified by all networks. Network with most exclusively correct predictions is in majority of replications densenet. Densenet annd resnet have in majority of replications most common correct predictions amongs the pairs of networks.

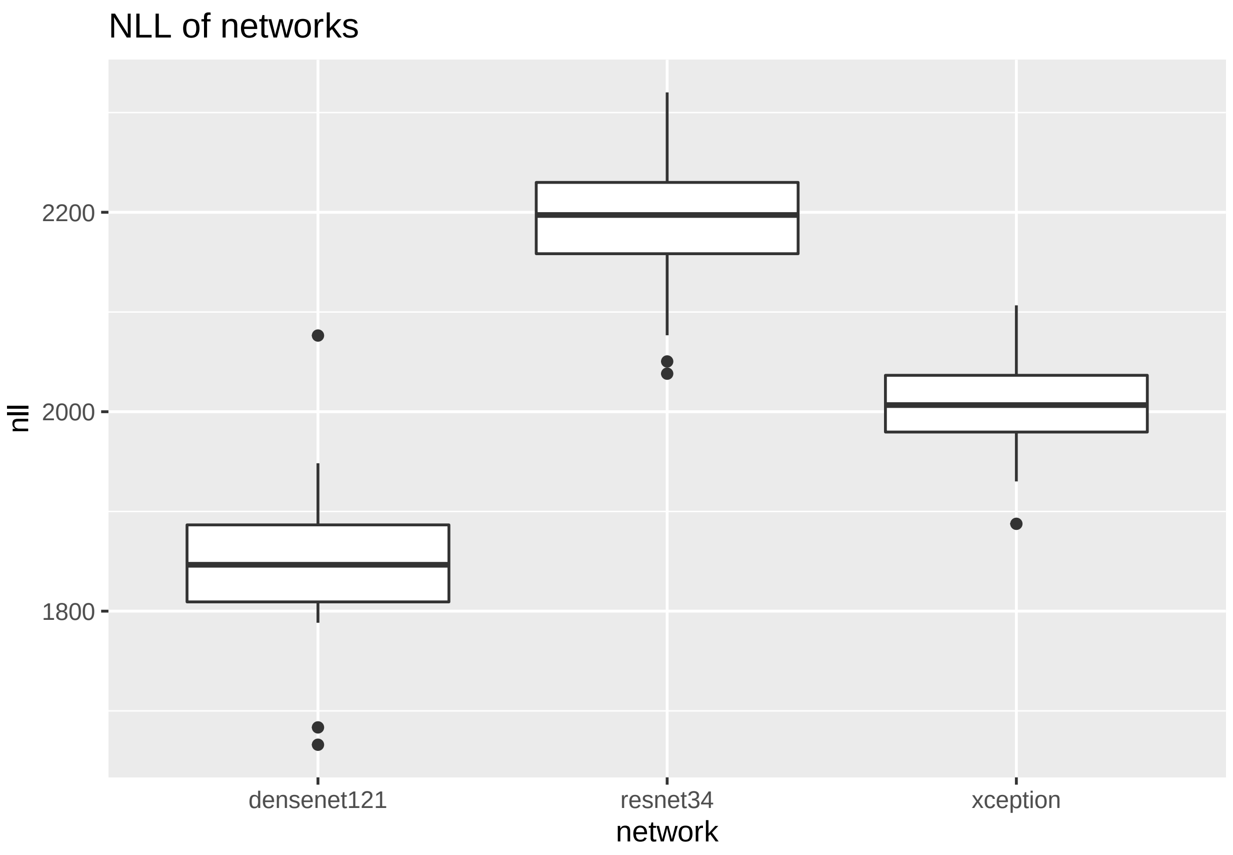
For clearer visualization, we will plot just the predicted probability of the correct class for all the methods.

```
preds <- nets_outputs$test_outputs
for (ri in repls + 1)
{
  for (net_i in seq_along(nets_outputs[["networks"]]))
  {
    preds[ri, net_i, ,] <- softmax(preds[ri, net_i, , ])
  }
}
nets_test_cor_probs <- gather(preds, 1 + nets_outputs$test_labels[1, ], 3, 4)
nets_test_cor_probs <- melt(nets_test_cor_probs)
nets_test_cor_probs <- nets_test_cor_probs[, c(-3, -4)]
names(nets_test_cor_probs) <- c("replication", "network", "prediction")
nets_test_cor_probs$network <- as.factor(nets_test_cor_probs$network)
levels(nets_test_cor_probs$network) <- nets_outputs$networks
```

```
nets_cor_preds_histo <- ggplot(data=nets_test_cor_probs) + geom_histogram(mapping=aes(x=prediction), bi
  ggtitle("Histograms of predicted probability for the correct class") + facet_wrap(~network) + scale_y
nets_cor_preds_histo
```



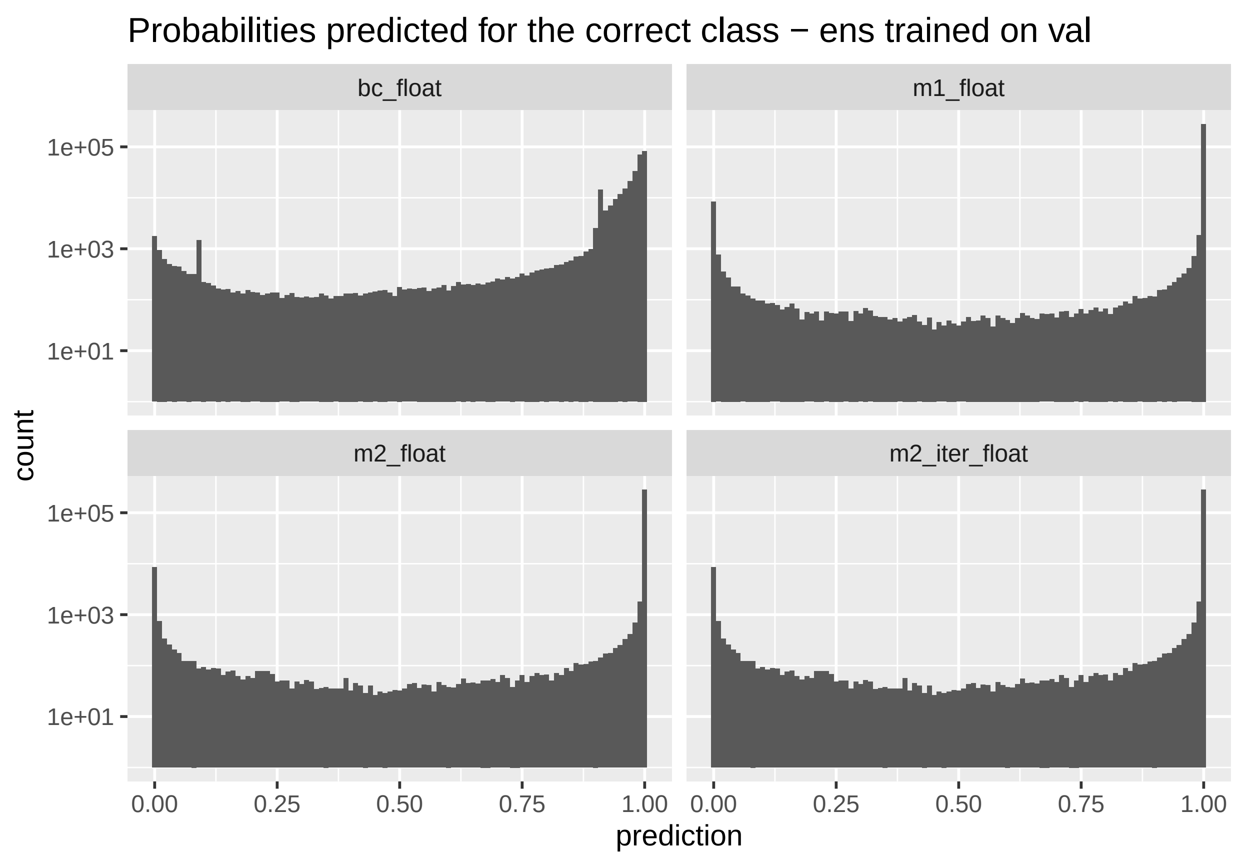
```
networks_nll <- ggplot(data=net_results) + geom_boxplot(mapping=aes(x=network, y=nll)) + ggtitle("NLL o
networks_nll
```



Networks nll seems to have good correspondence with lowest prediction probability column in the previous histograms.

```
val_ens_cor_probs <- gather(ens_outputs$val_training, 1 + nets_outputs$test_labels[1, ], 3, 4)
val_ens_cor_probs <- melt(val_ens_cor_probs)
val_ens_cor_probs <- val_ens_cor_probs[, c(-3, -4)]
names(val_ens_cor_probs) <- c("replication", "method", "prediction")
val_ens_cor_probs$method <- as.factor(val_ens_cor_probs$method)
levels(val_ens_cor_probs$method) <- ens_outputs$methods
```

```
val_ens_cor_preds_histo <- ggplot(data=val_ens_cor_probs) + geom_histogram(mapping=aes(x=prediction), b
val_ens_cor_preds_histo
```

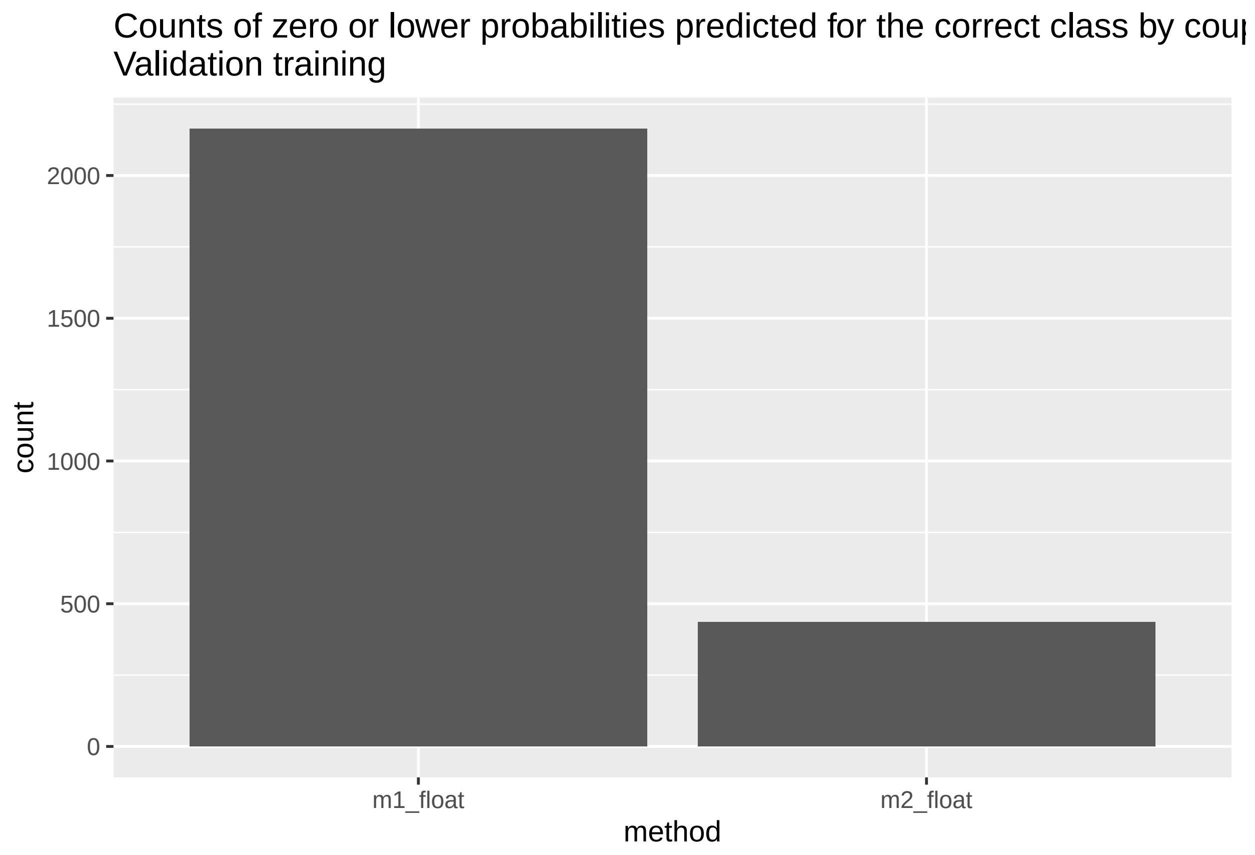


Coupling method bc produces far fewer probabilities falling into the lowest bin for the correct class than m1 and m2.

```
val_ens_zero_counts <- ggplot(data=val_ens_cor_probs[val_ens_cor_probs$prediction <= 0, ]) + geom_histog
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

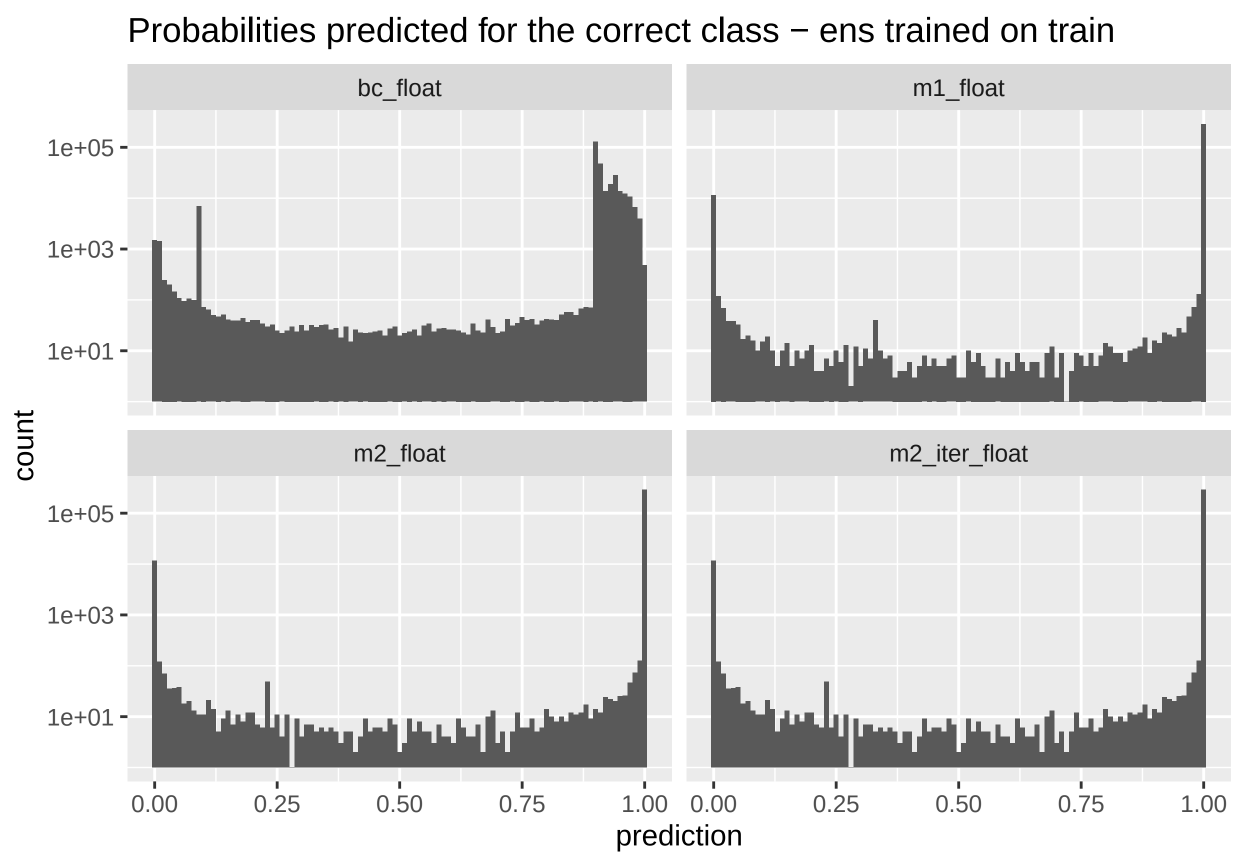
```
val_ens_zero_counts
```



m2_iter and bc didn't produce any zero probability outputs in this replication.

```
train_ens_cor_probs <- gather(ens_outputs$train_training, 1 + nets_outputs$test_labels[1, ], 3, 4)
train_ens_cor_probs <- melt(train_ens_cor_probs)
train_ens_cor_probs <- train_ens_cor_probs[, c(-3, -4)]
names(train_ens_cor_probs) <- c("replication", "method", "prediction")
train_ens_cor_probs$method <- as.factor(train_ens_cor_probs$method)
levels(train_ens_cor_probs$method) <- ens_outputs$methods
```

```
train_ens_cor_preds_histo <- ggplot(data=train_ens_cor_probs) + geom_histogram(mapping=aes(x=prediction
train_ens_cor_preds_histo
```

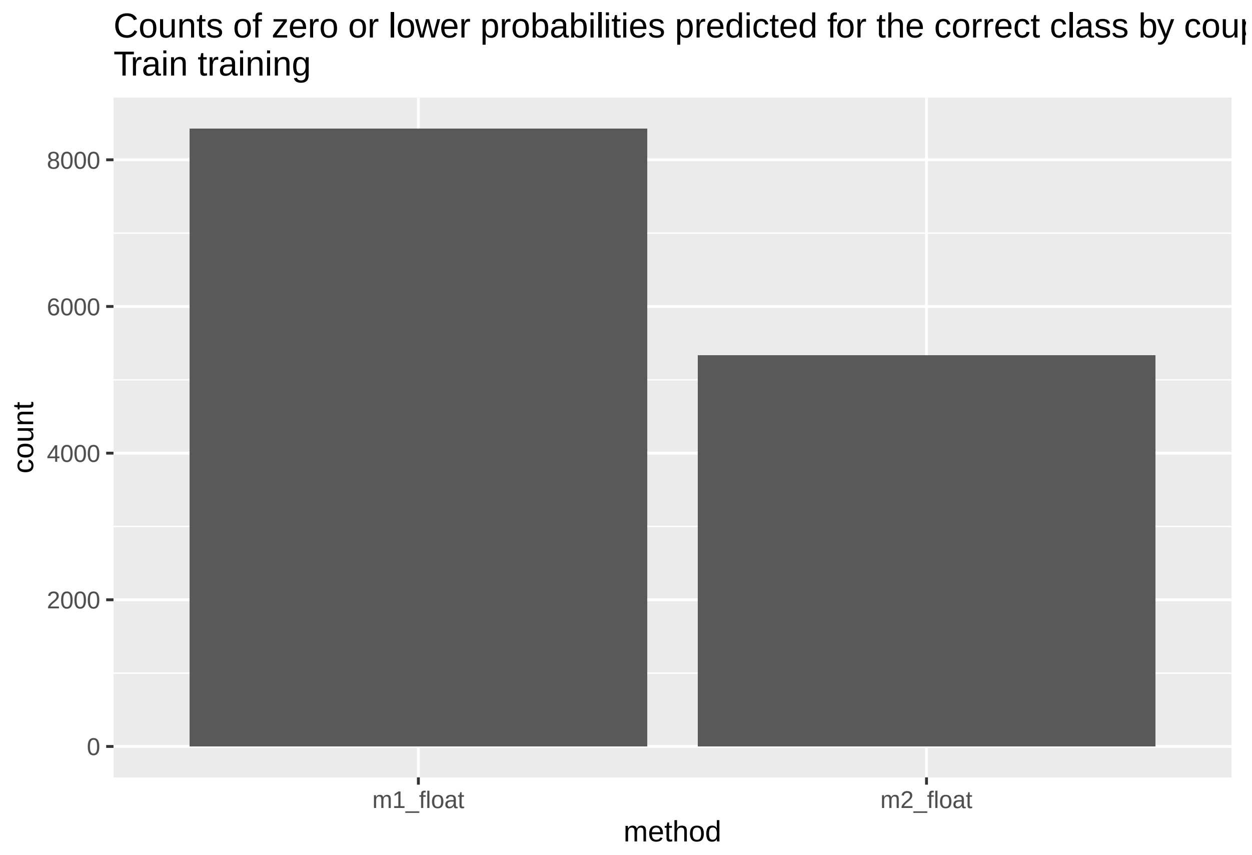


Also in this case, coupling method bc produces far fewer probabilities falling into the lowest bin for the correct class than m1 and m2.

```
train_ens_zero_counts <- ggplot(data=train_ens_cor_probs[train_ens_cor_probs$prediction <= 0, ]) + geom_
```

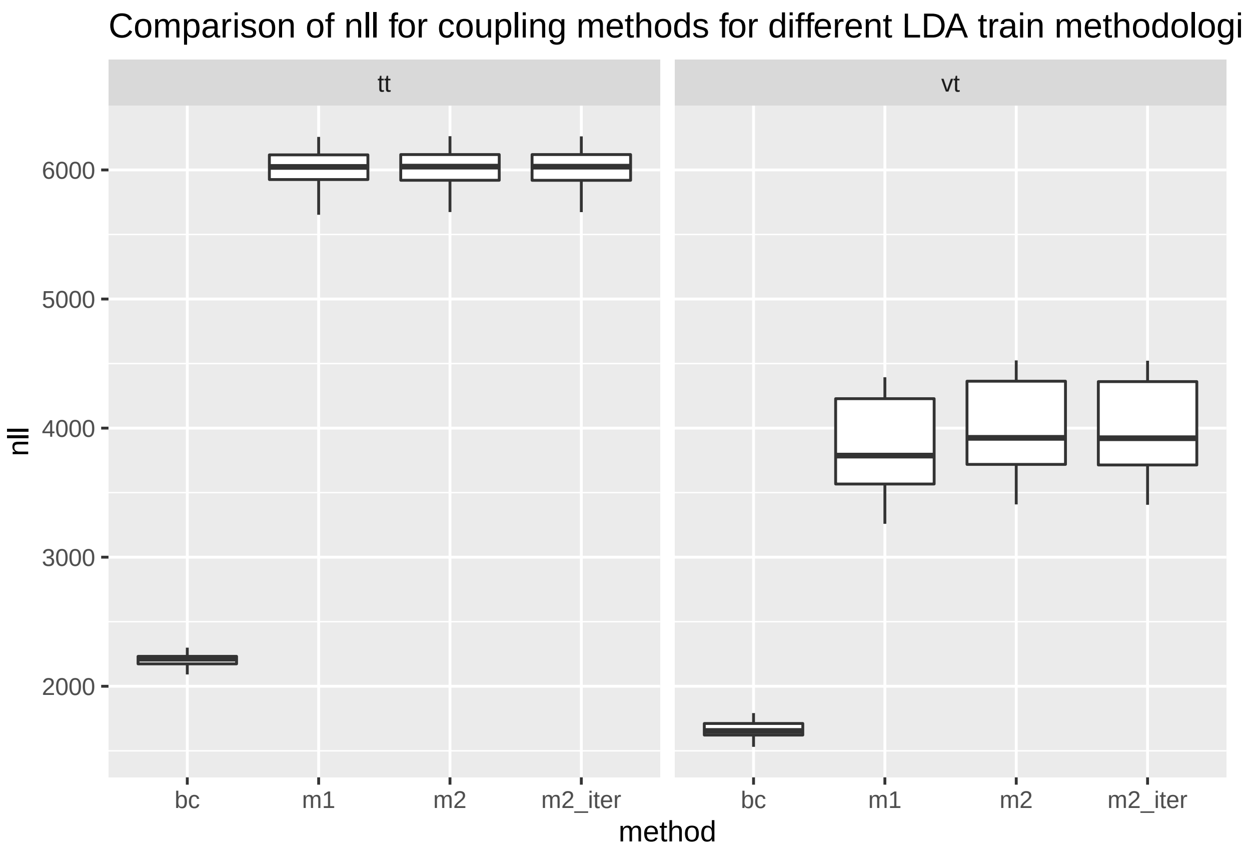
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
train_ens_zero_counts
```



m2_iter and bc didn't produce any zero probability outputs in this replication.

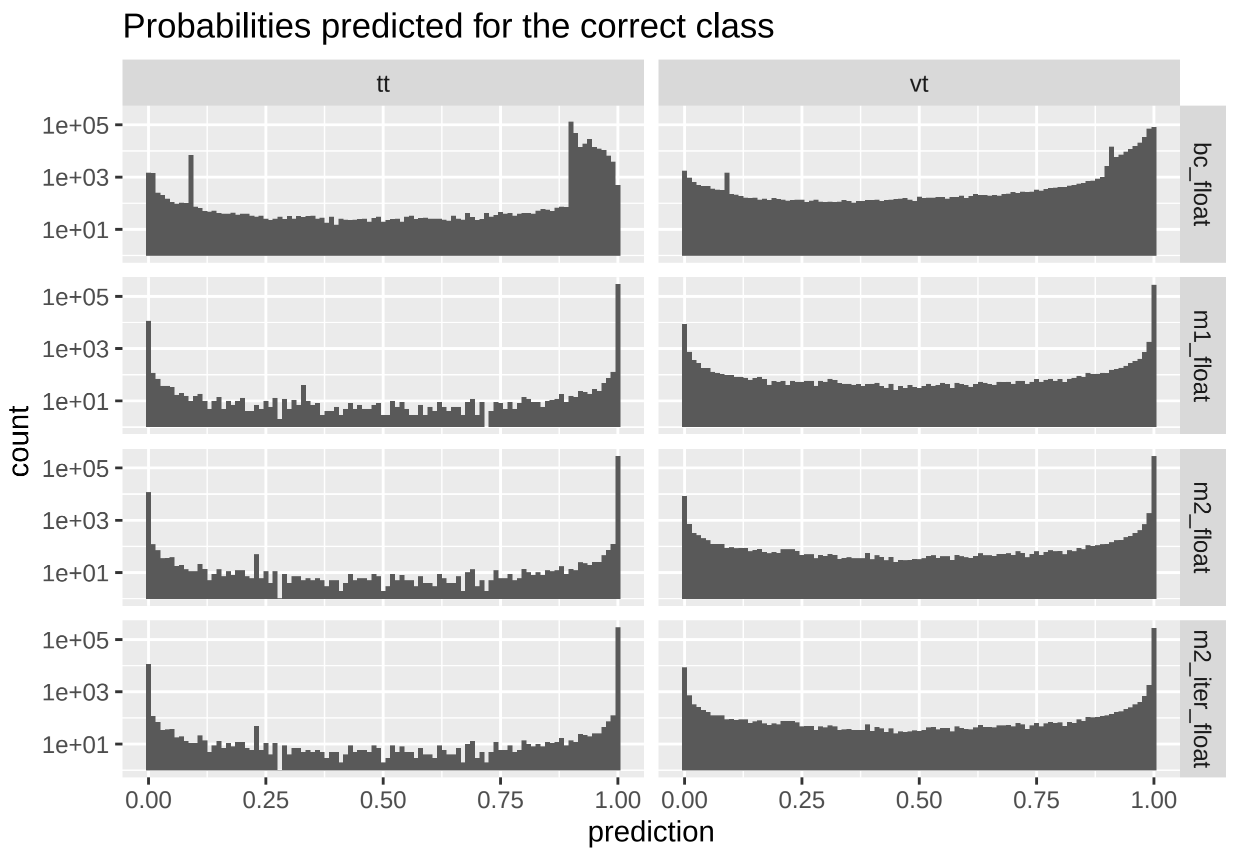
```
val_ens_nll <- ggplot(data=ens_results) + geom_boxplot(mapping=aes(x=method, y=nll)) + facet_wrap(~trai
  ggtitle("Comparison of nll for coupling methods for different LDA train methodologies")
val_ens_nll
```



When LDA is trained on training data, bc method has clearly superior nll, however in the case of training LDA on validation data, other coupling methods perform similarly, or better.

```
val_ens_cor_probs$train_type <- "vt"
train_ens_cor_probs$train_type <- "tt"
ens_cor_probs <- rbind(val_ens_cor_probs, train_ens_cor_probs)
```

```
ens_cor_preds_histo <- ggplot(data=ens_cor_probs) + geom_histogram(mapping=aes(x=prediction), binwidth=
ens_cor_preds_histo
```



Bayes covariant coupling method produces more uniformly distributed predictions than methods m1 and m2. Also, there is a big difference in each method between ensemble trained on validation and ensemble trained on train set. Ensembles trained on validation set produce generally more uniformly distributed predictions. However, ensembles trained on training set attain statistically significantly higher accuracy.

```
ens_Rs <- load_R_matrices(base_dir, repls)
```

```
df_val_Rs <- melt(ens_Rs$val_training)
names(df_val_Rs) <- c("replication", "precision", "sample", "class1", "class2", "prob")
class <- nets_outputs$test_labels[1, df_val_Rs$sample] + 1
df_val_Rs$class <- as.factor(class)
df_val_Rs[,c("class1", "class2")] <- lapply(df_val_Rs[,c("class1", "class2")], factor)
val_class_mean_Rs <- df_val_Rs %>% group_by(precision, class1, class2, class) %>% summarise(prob=mean(pr
```

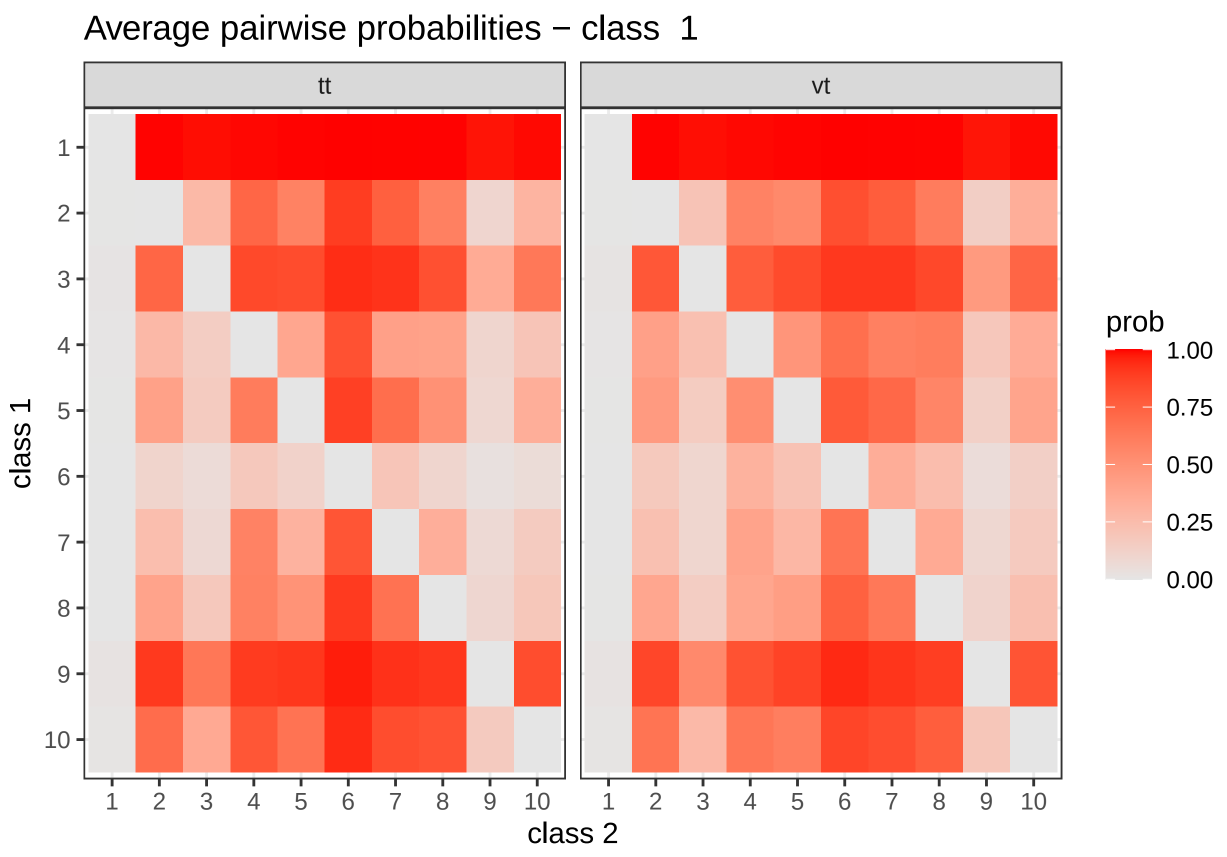
```
## ‘summarise()‘ has grouped output by ’precision’, ’class1’, ’class2’. You can override using the ‘.gro
```

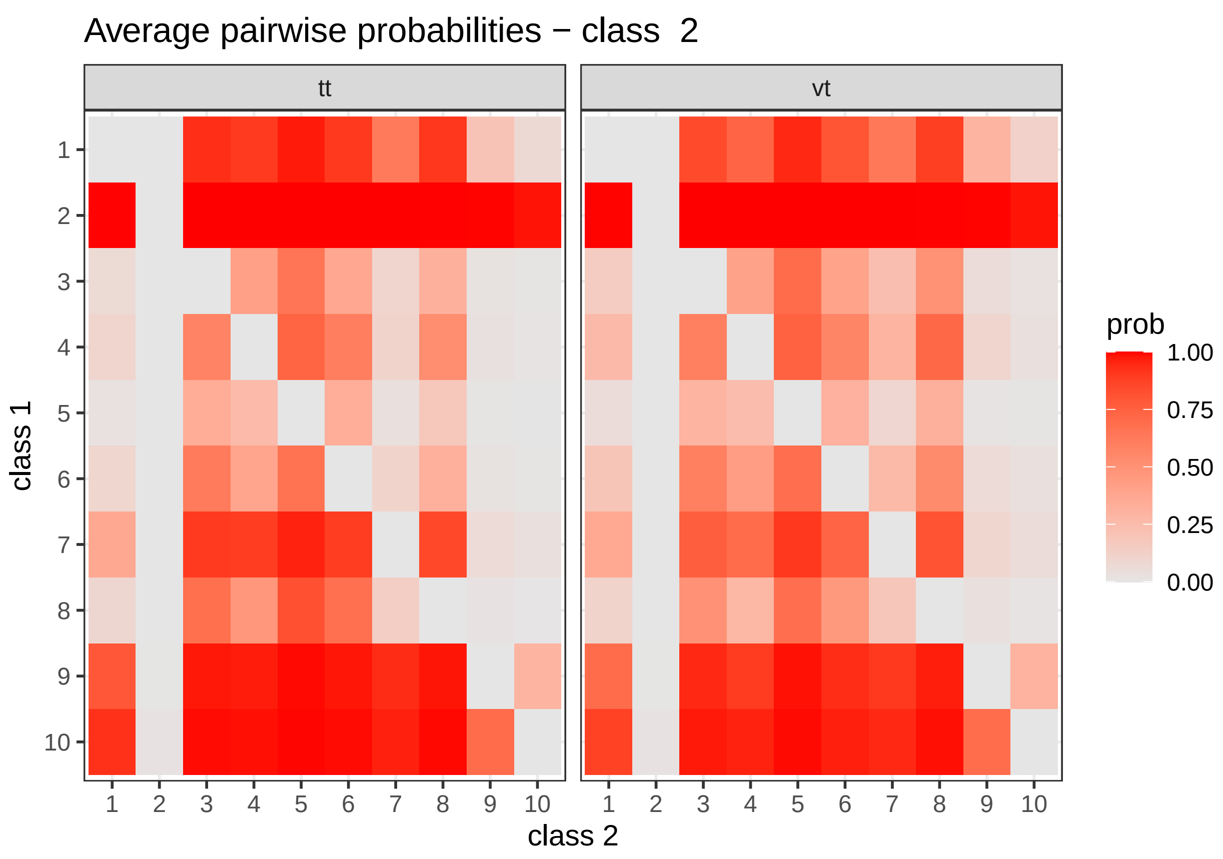
```
df_train_Rs <- melt(ens_Rs$train_training)
names(df_train_Rs) <- c("replication", "precision", "sample", "class1", "class2", "prob")
class <- nets_outputs$test_labels[1, df_train_Rs$sample] + 1
df_train_Rs$class <- as.factor(class)
df_train_Rs[,c("class1", "class2")] <- lapply(df_train_Rs[,c("class1", "class2")], factor)
train_class_mean_Rs <- df_train_Rs %>% group_by(precision, class1, class2, class) %>% summarise(prob=mea
```

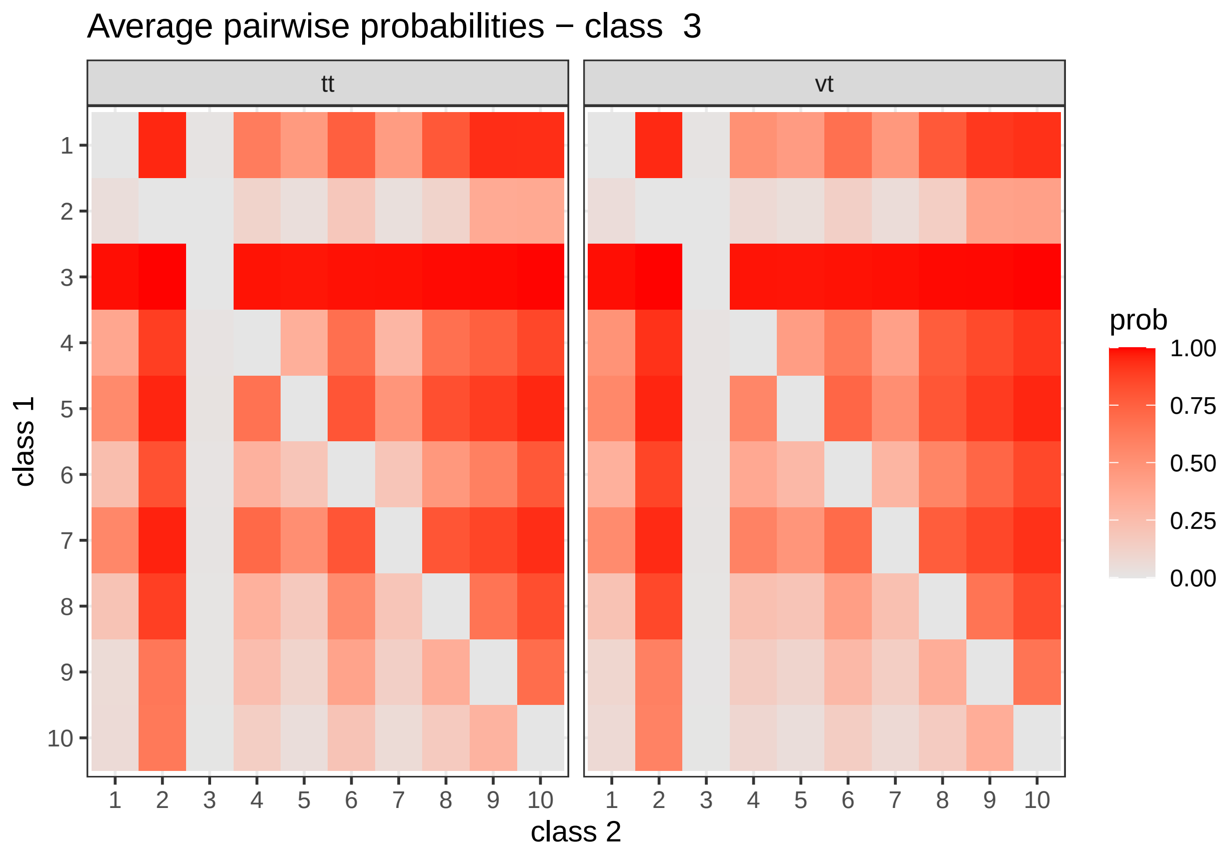
```
## ‘summarise()‘ has grouped output by ’precision’, ’class1’, ’class2’. You can override using the ‘.gro
```

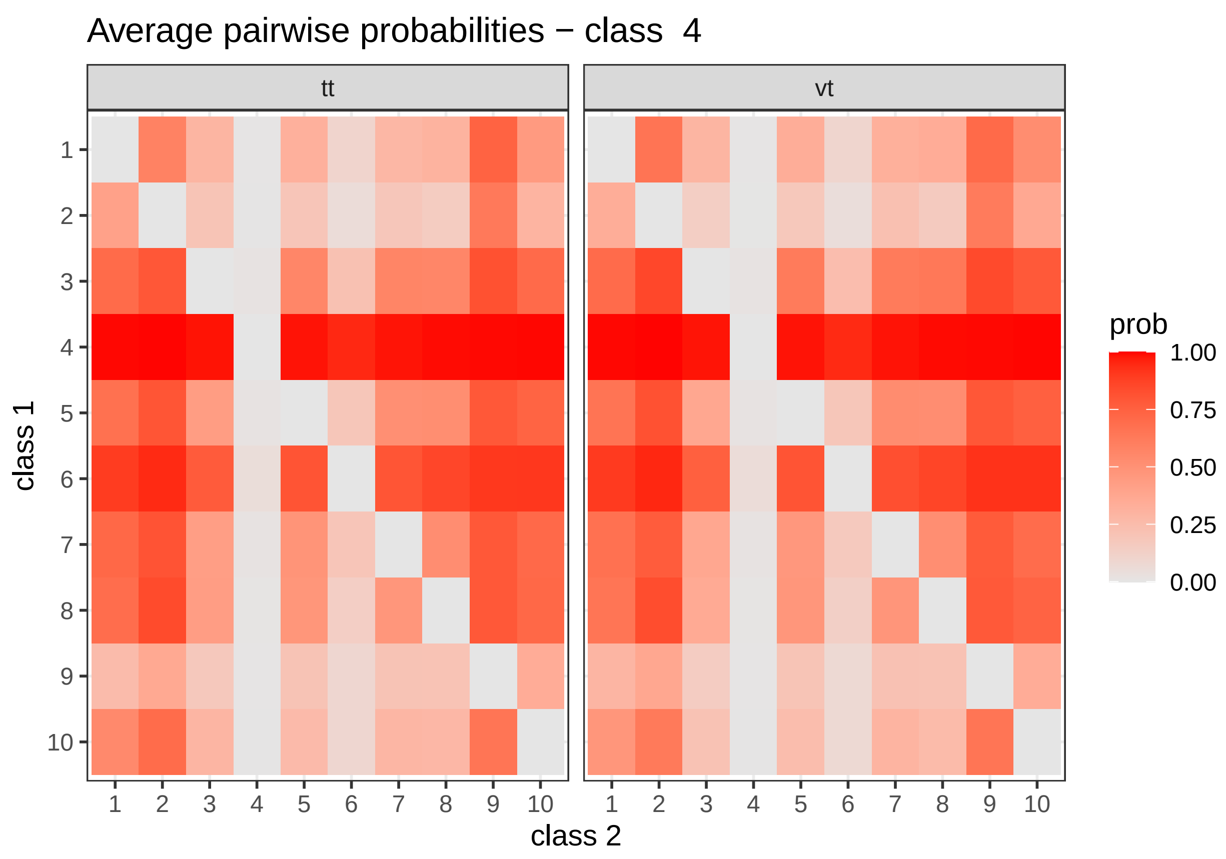
```
val_class_mean_Rs$train_type <- "vt"
train_class_mean_Rs$train_type <- "tt"
class_mean_Rs <- rbind(val_class_mean_Rs, train_class_mean_Rs)

df_aggr_Rs_diff <- class_mean_Rs %>% pivot_wider(names_from = train_type, values_from = prob) %>% mutat
```

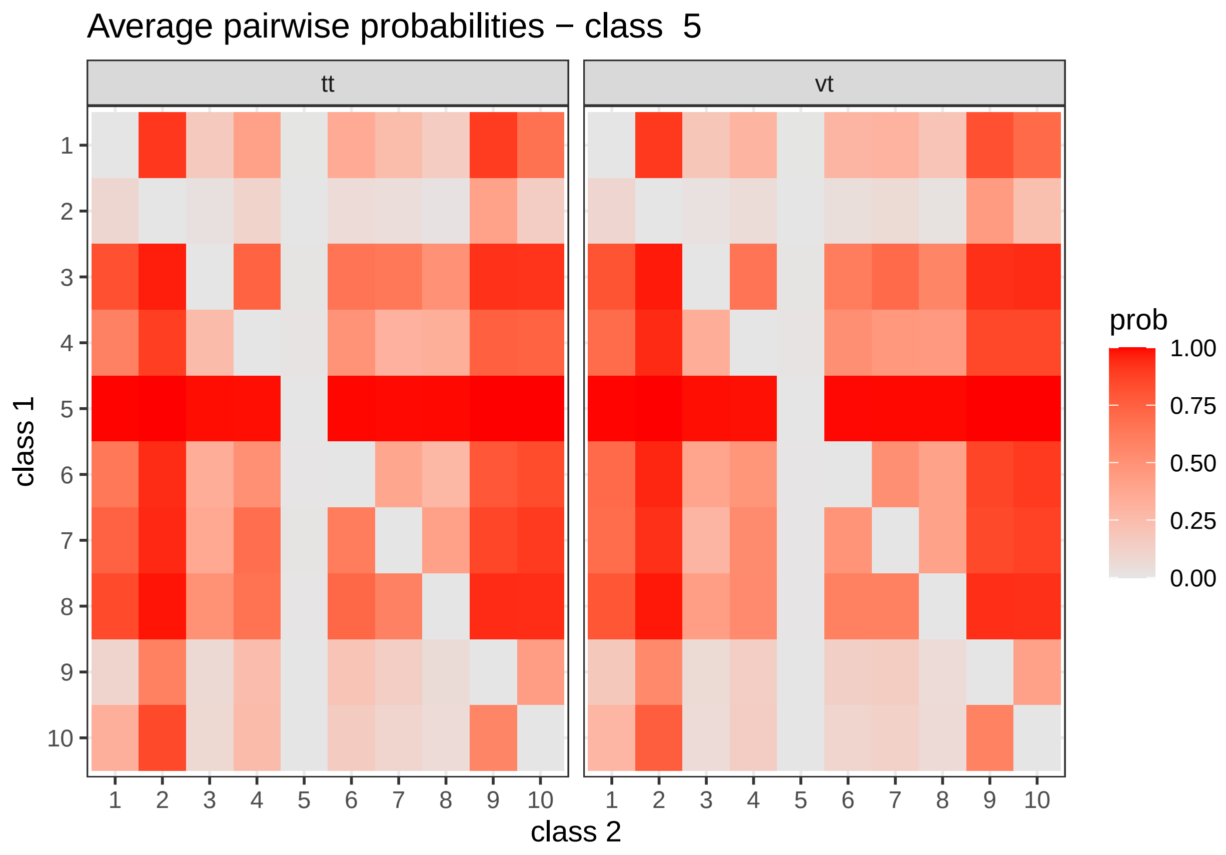
```
for (cls in 1:classes)
{
  cur_class_Rs <- class_mean_Rs %>% filter(class == cls)
  plot_cls <-  ggplot(cur_class_Rs, aes(x = class2, y = class1)) +
    geom_raster(aes(fill=prob)) +
    facet_wrap(~train_type) +
    scale_fill_gradient(low="grey90", high="red", limits=c(0, 1)) +
    scale_y_discrete(limits=rev) +
    labs(x="class 2", y="class 1", title=paste("Average pairwise probabilities - class ", cls)) +
    theme_bw()

  print(plot_cls)
}
```

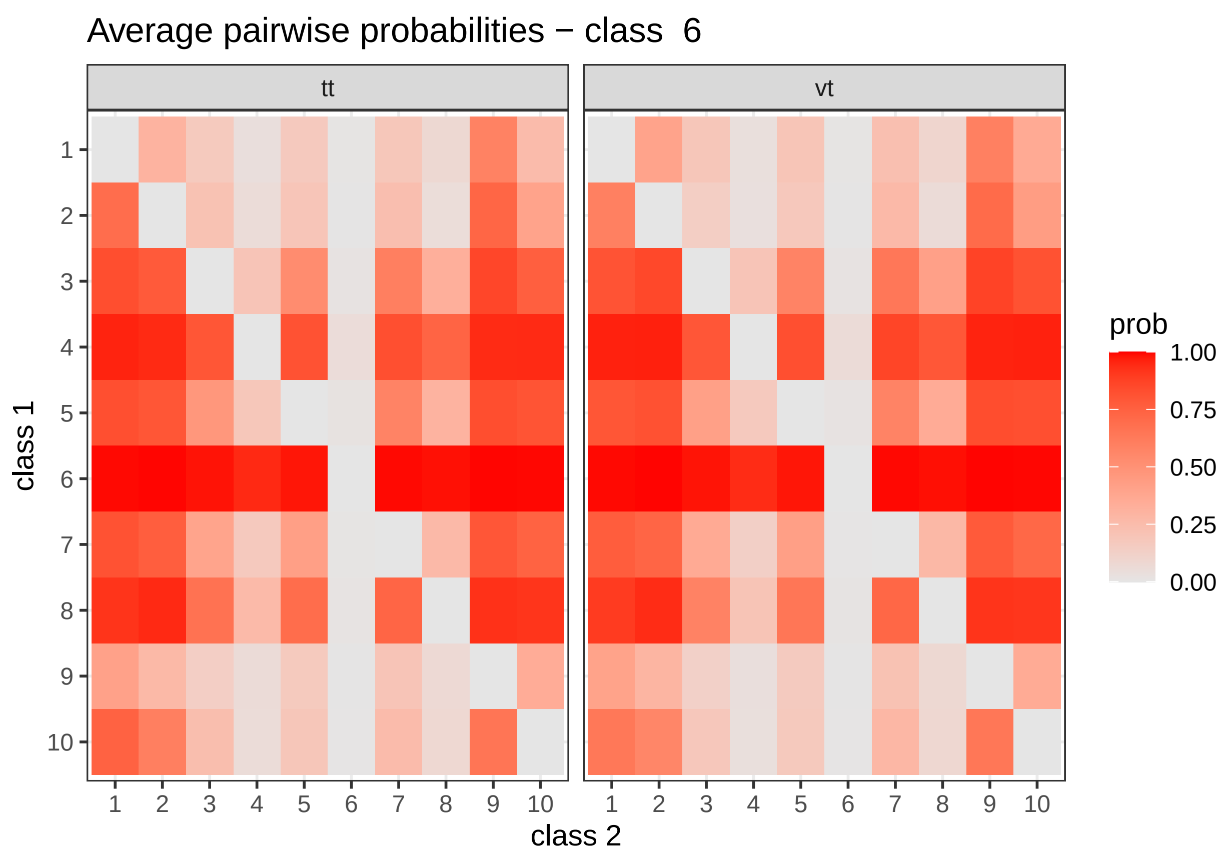


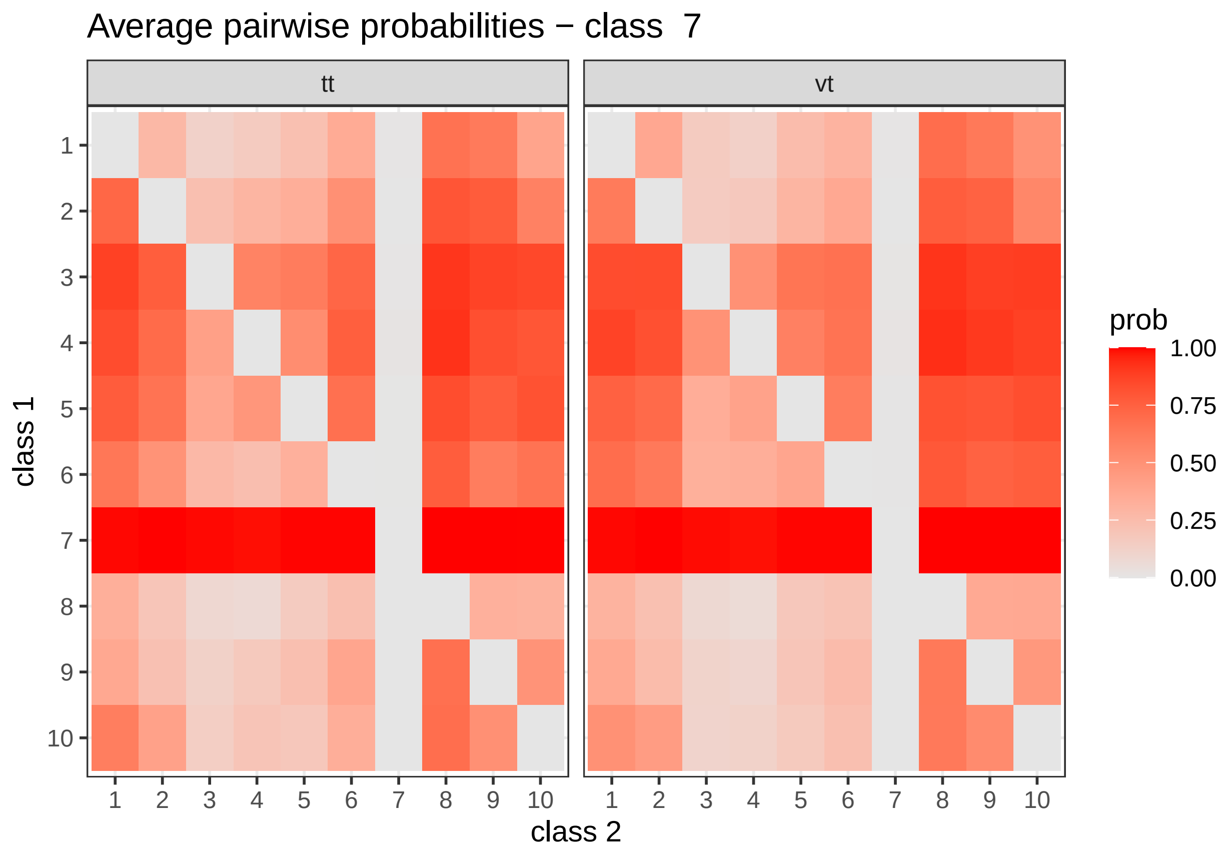

Average pairwise probabilities − class 2
tt
vt
class 1
class 2
prob
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

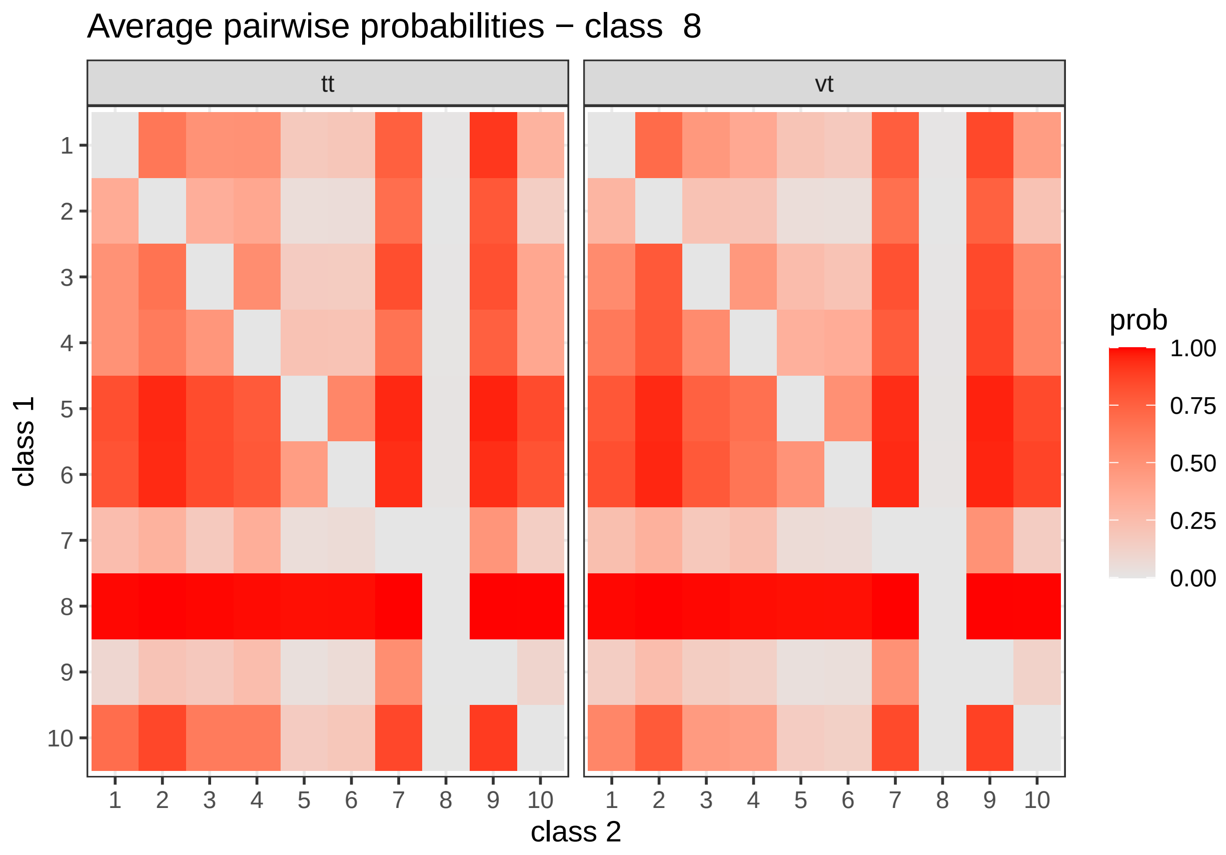

Average pairwise probabilities − class 3
tt
vt
class 1
class 2
prob
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

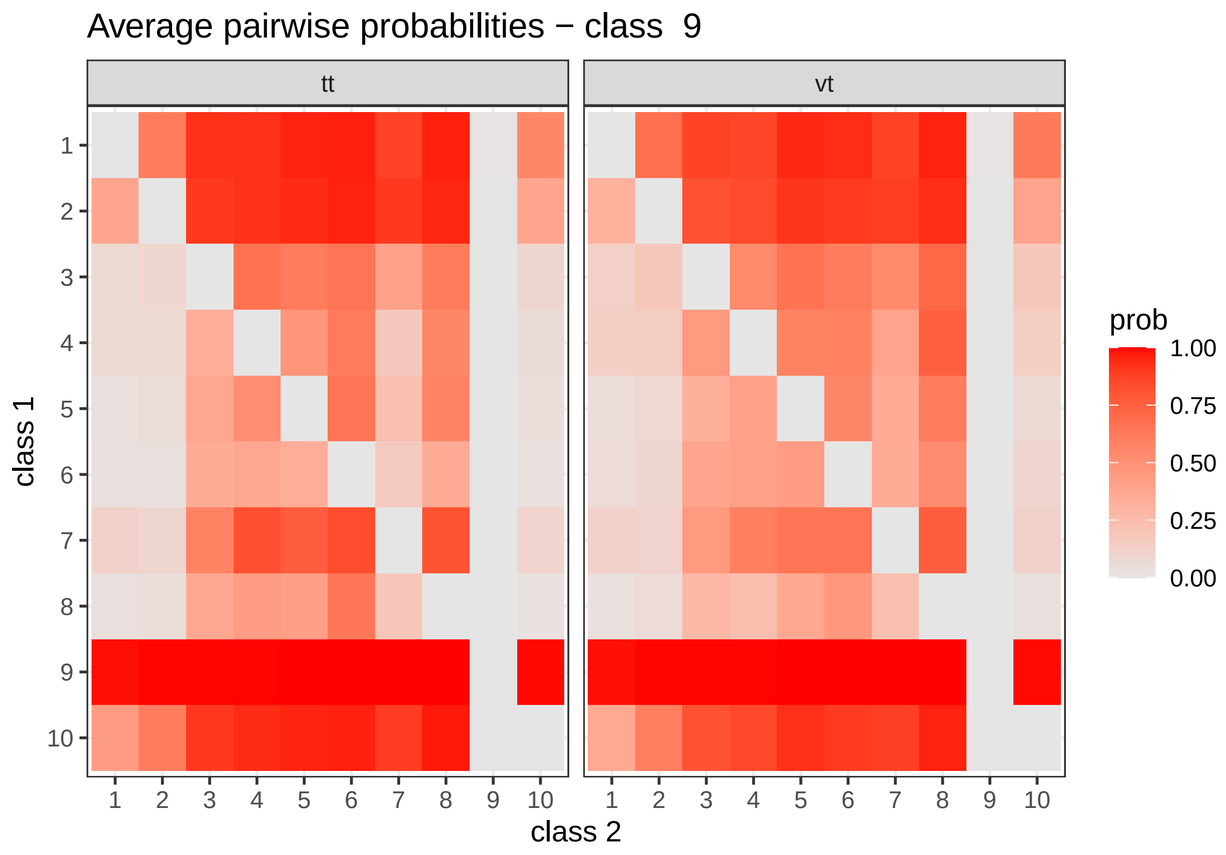

Average pairwise probabilities − class 4
tt
vt
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
1.00
0.75
0.50
0.25
0.00

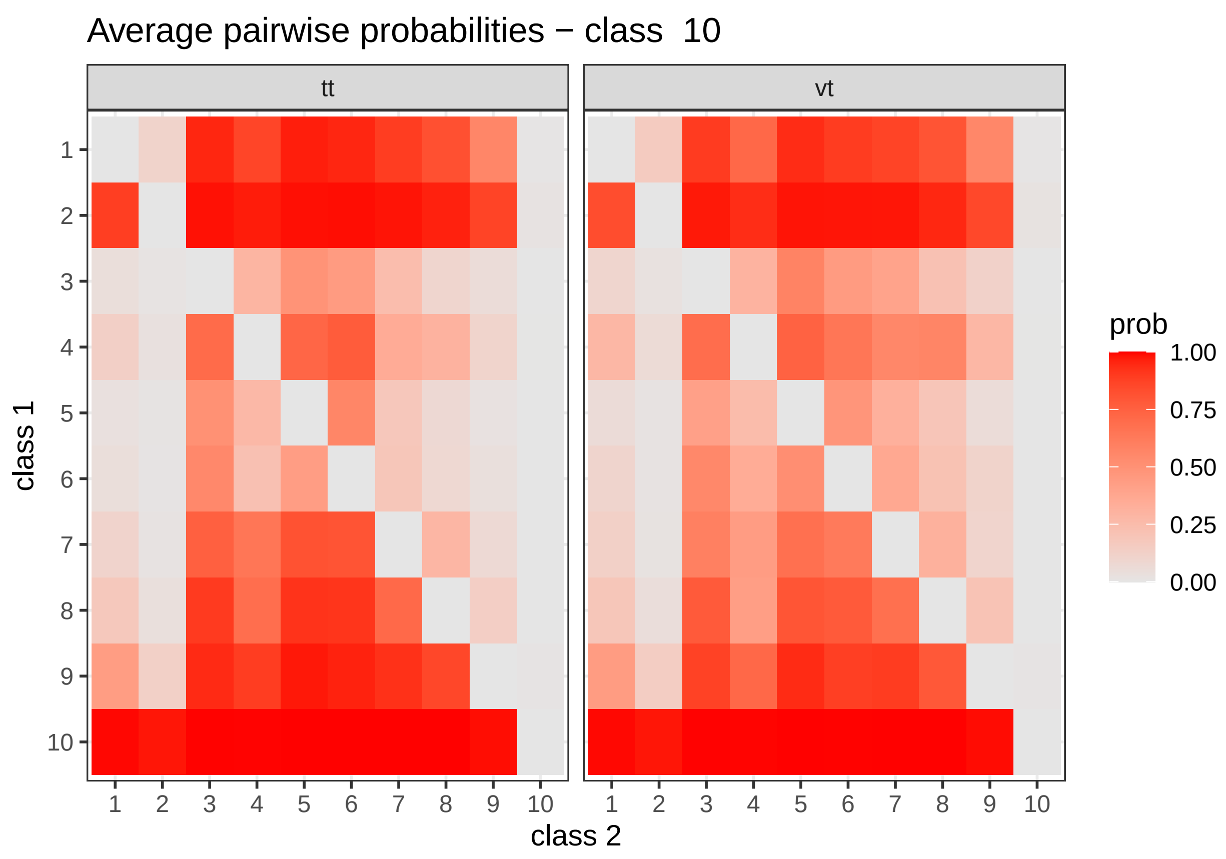

Average pairwise probabilities − class 5
tt
vt
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
1.00
0.75
0.50
0.25
0.00


Average pairwise probabilities − class 6
tt
vt
class 1
class 2
prob
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

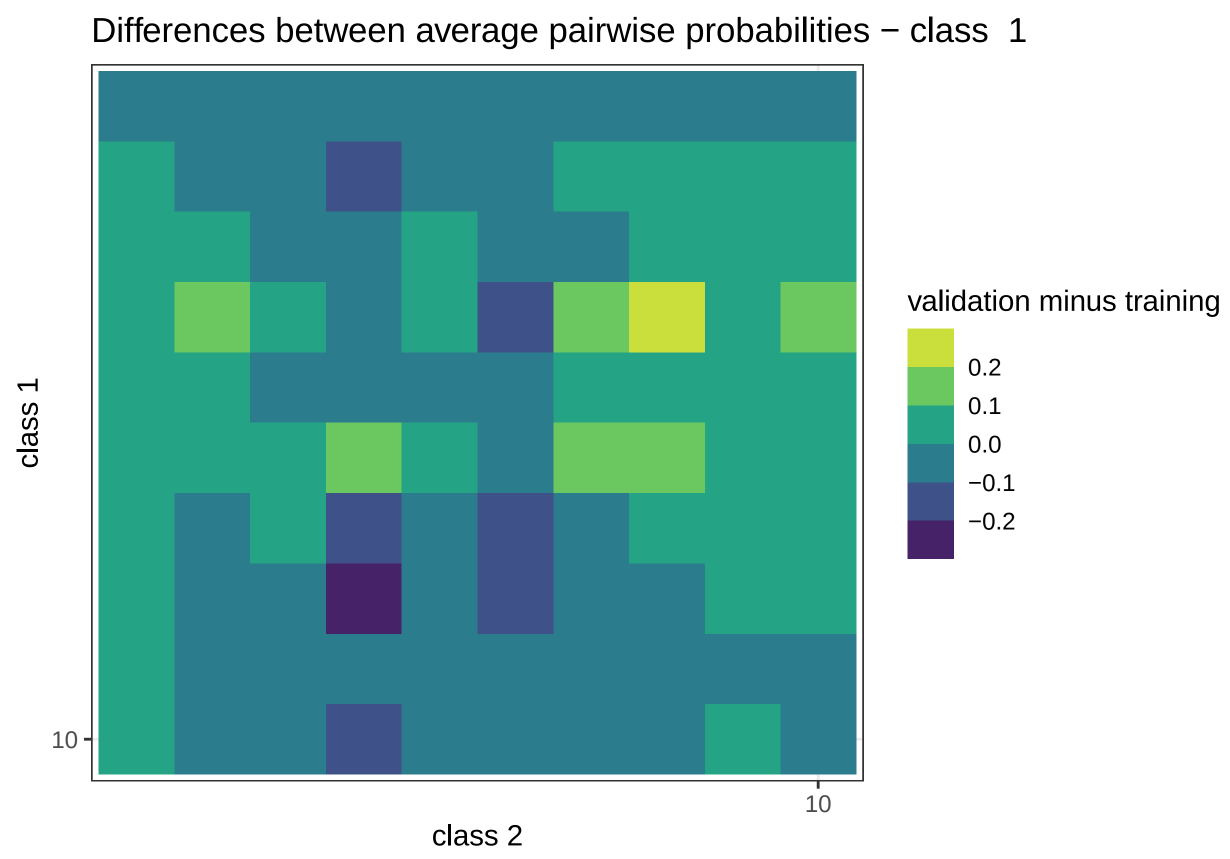

Average pairwise probabilities − class 7
tt
vt
class 1
class 2
prob
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

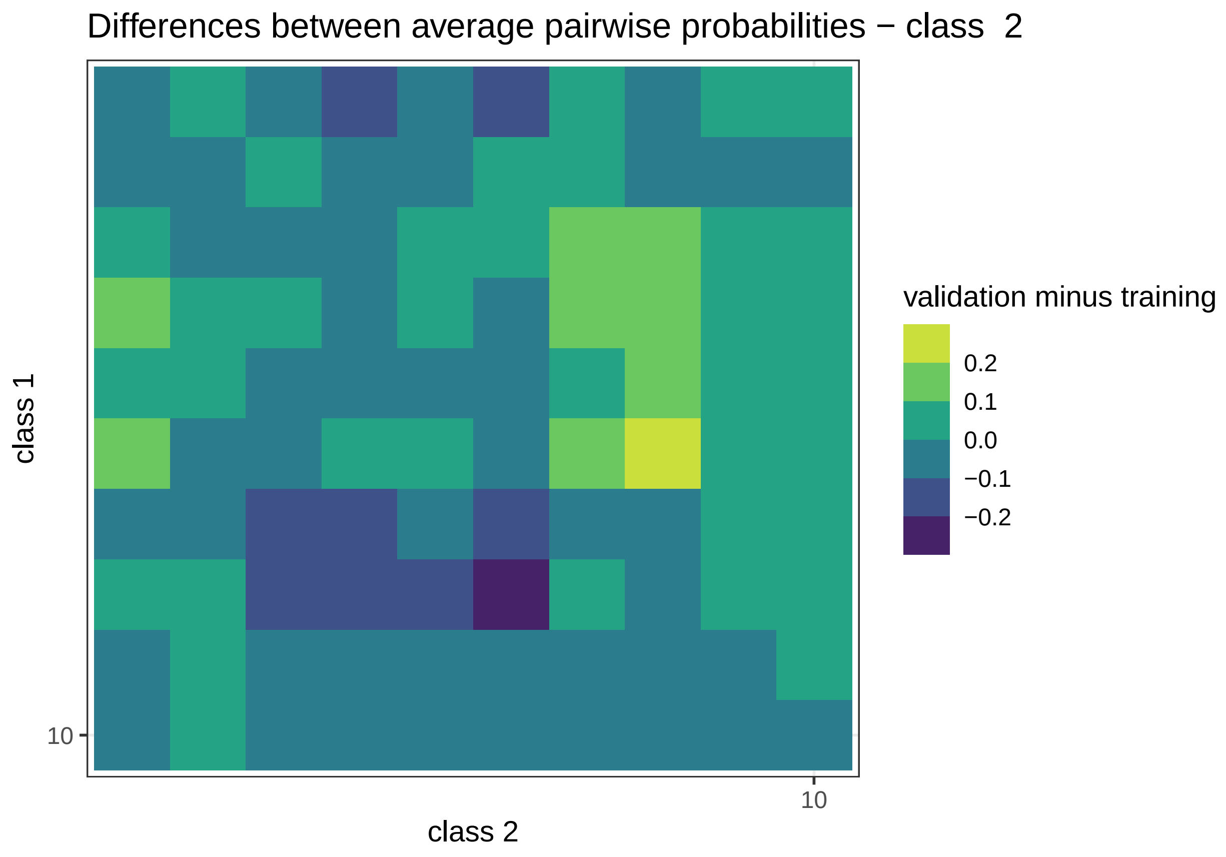

Average pairwise probabilities − class 8
tt
vt
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
1.00
0.75
0.50
0.25
0.00

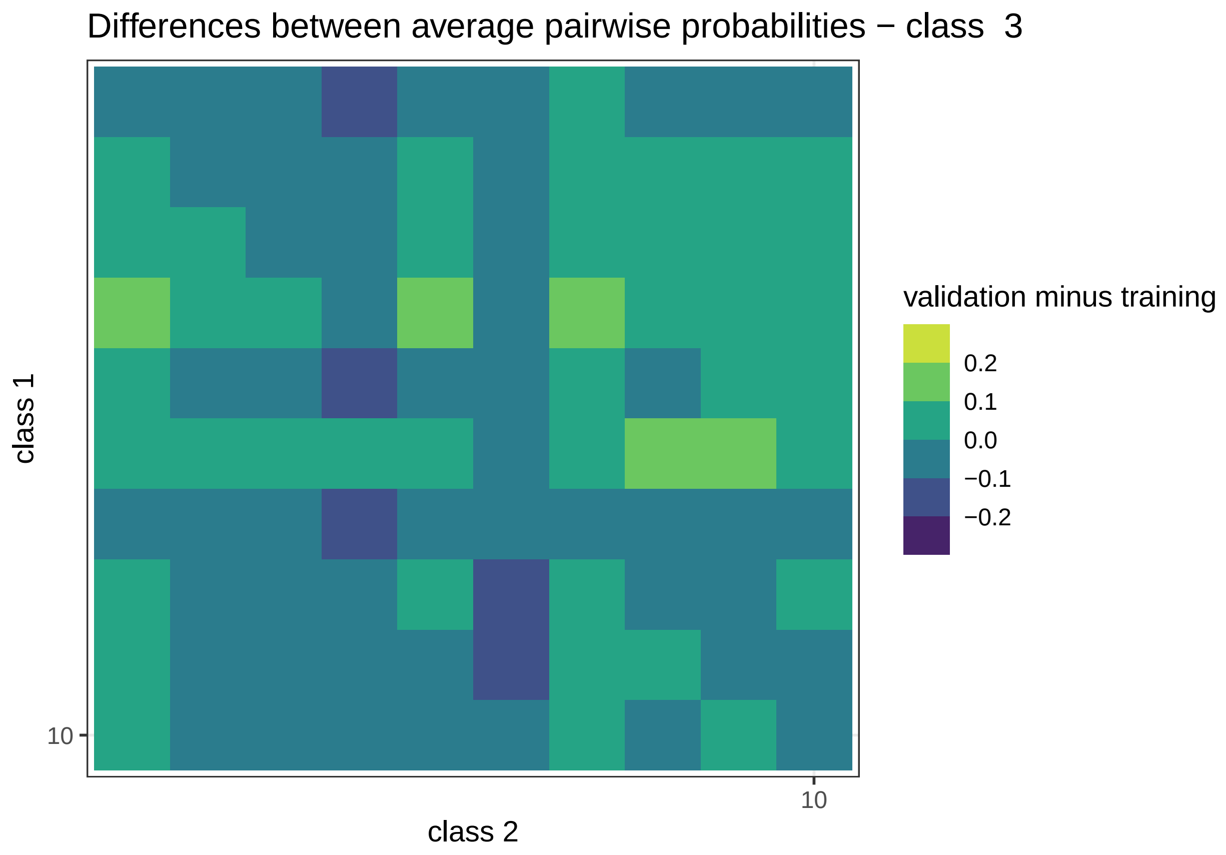

Average pairwise probabilities − class 9
tt
vt
class 1
class 2
prob
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

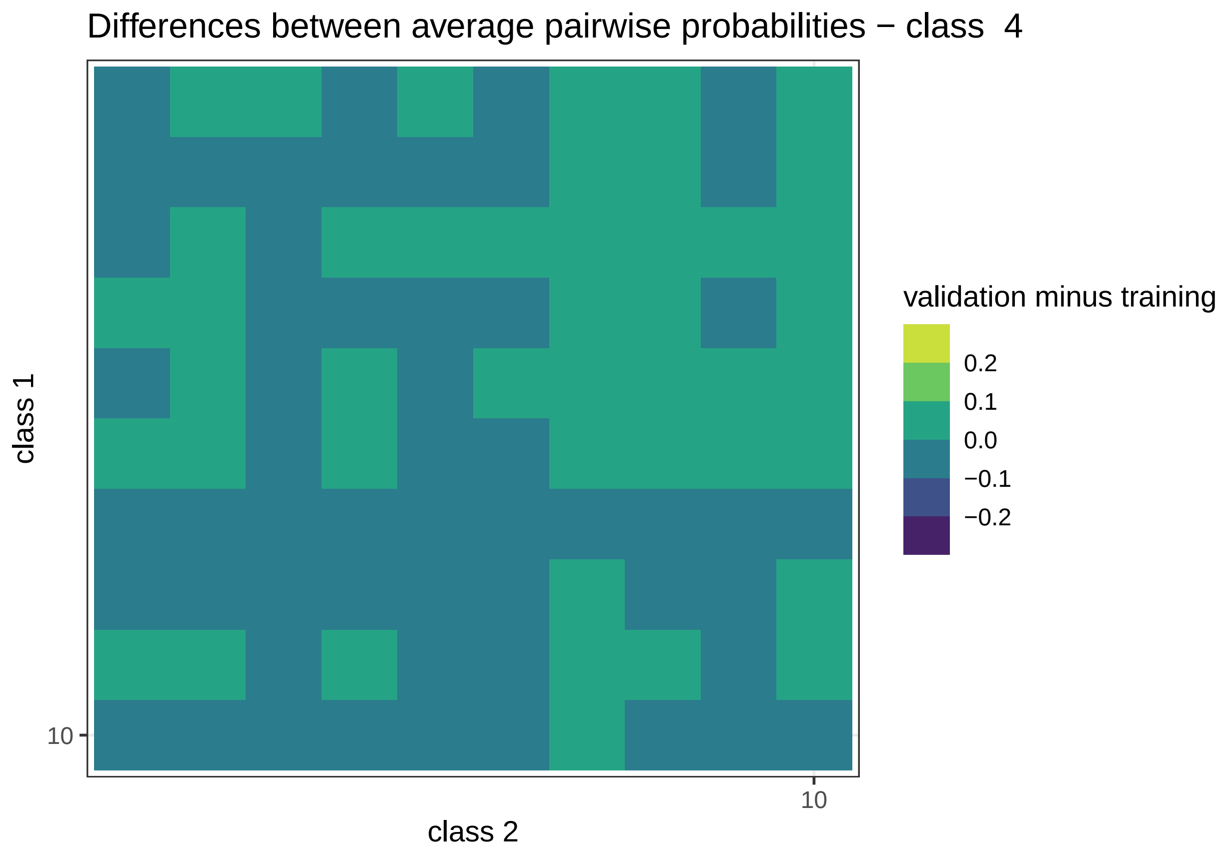


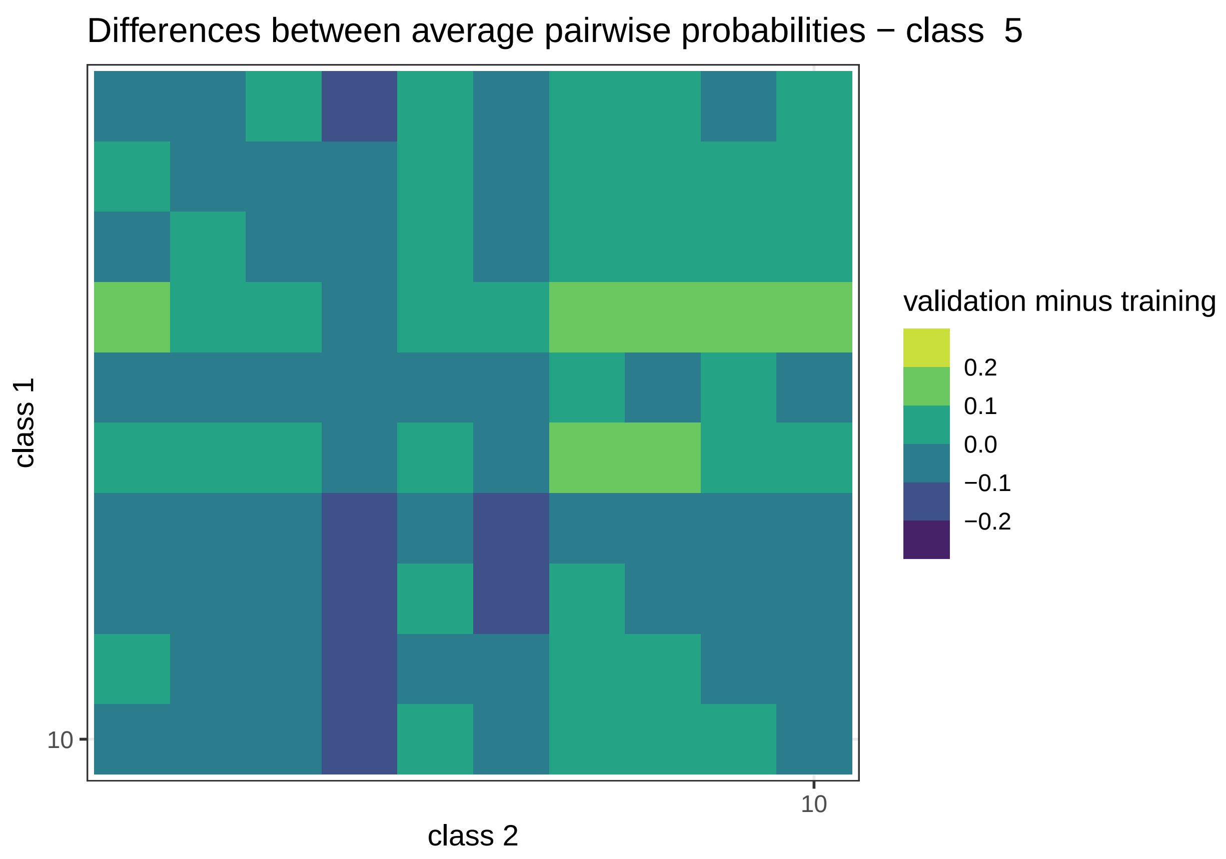
```r
for (cls in 1:classes)
{
  cur_class_Rs <- df_aggr_Rs_diff %>% filter(class == cls)
  plot_cls <-  ggplot(cur_class_Rs, aes(x = class2, y = class1)) +
    geom_raster(aes(fill=val_min_train)) +
    scale_fill_binned(type="viridis", limits=c(-0.3, 0.3), name="validation minus training") +
    scale_y_discrete(limits=rev, breaks=seq(0, classes, 10)) +
    scale_x_discrete(breaks=seq(0, classes, 10)) +
    labs(x="class 2", y="class 1", title=paste("Differences between average pairwise probabilities - cl
    theme_bw()

  print(plot_cls)
}
```

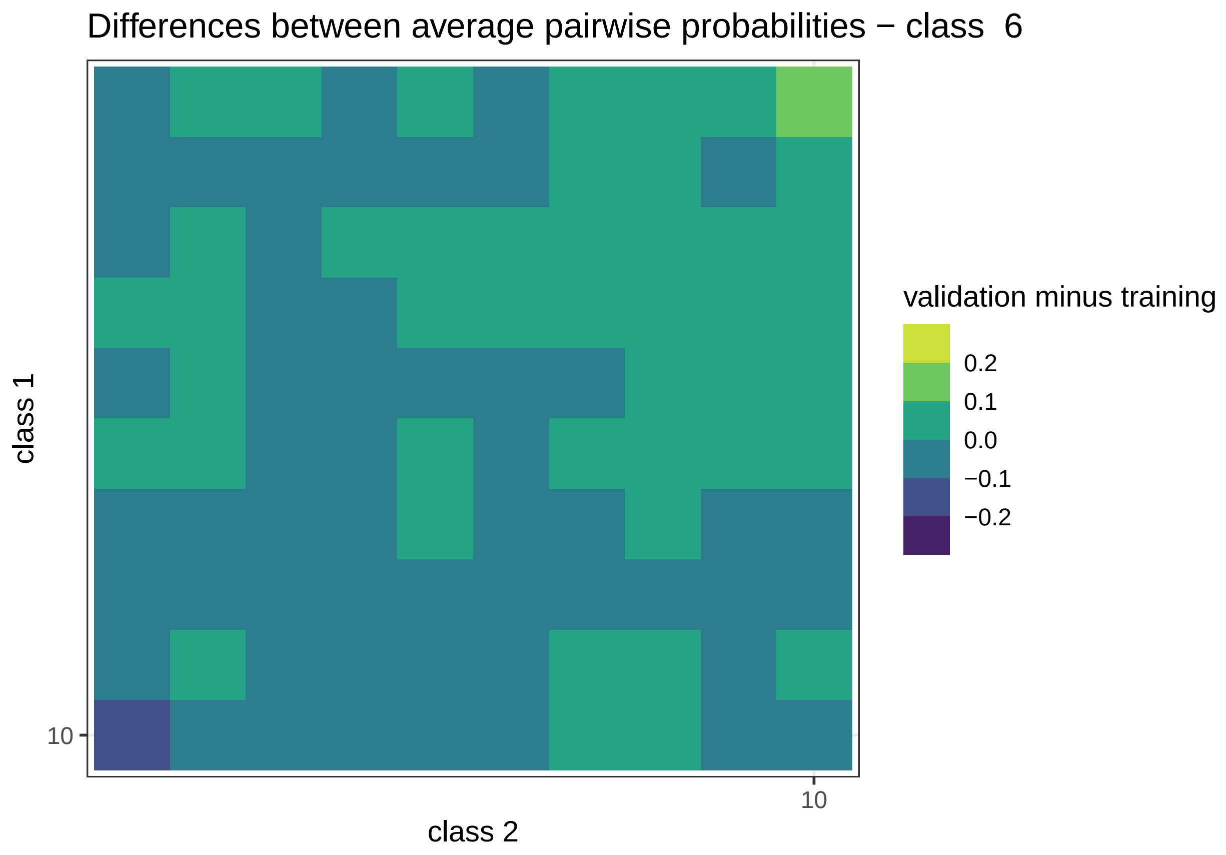

Differences between average pairwise probabilities − class 1
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

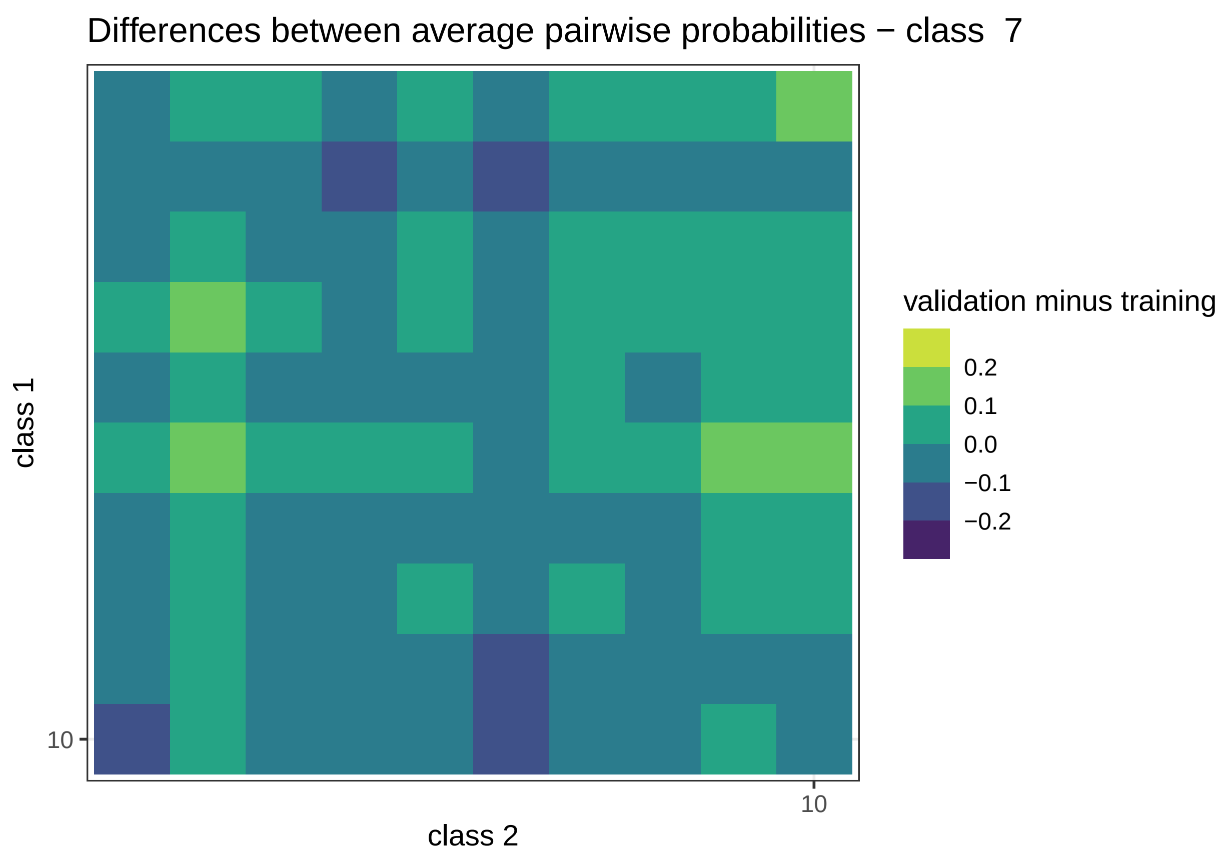

Differences between average pairwise probabilities − class 2
validation minus training
0.2
0.1
0.0
−0.1
−0.2
class 1
class 2
10
10

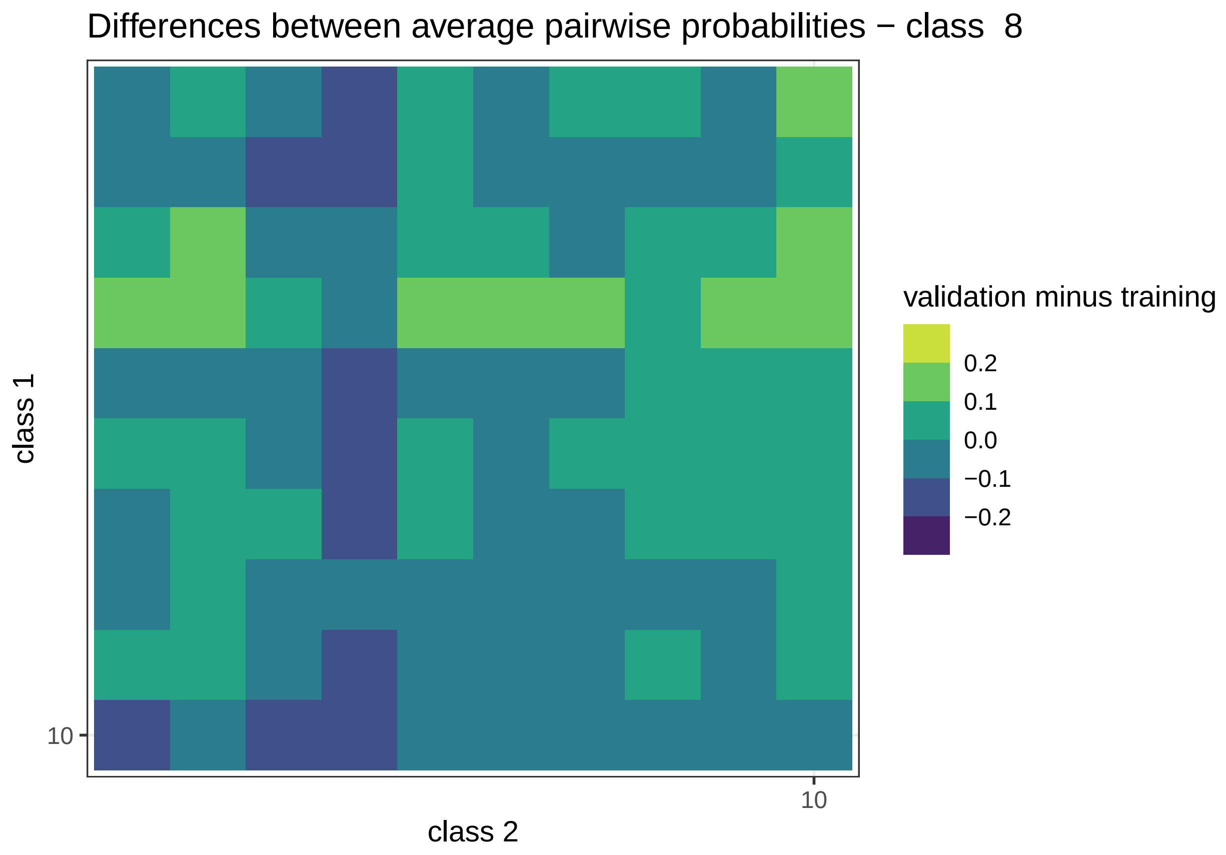

Differences between average pairwise probabilities − class 3
class 1
10
class 2
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2


Differences between average pairwise probabilities − class 4
validation minus training
0.2
0.1
0.0
−0.1
−0.2
class 1
10
class 2
10

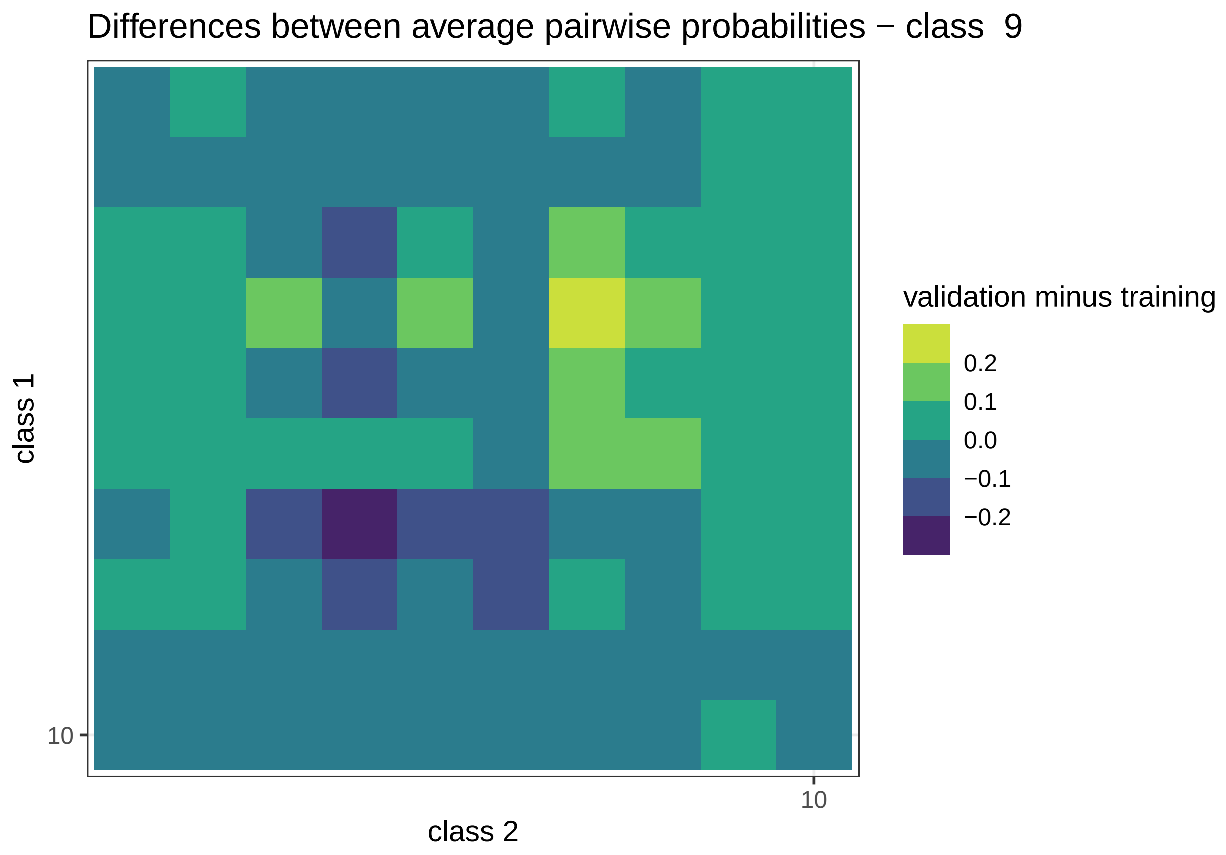

Differences between average pairwise probabilities − class 5
class 1
10
class 2
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

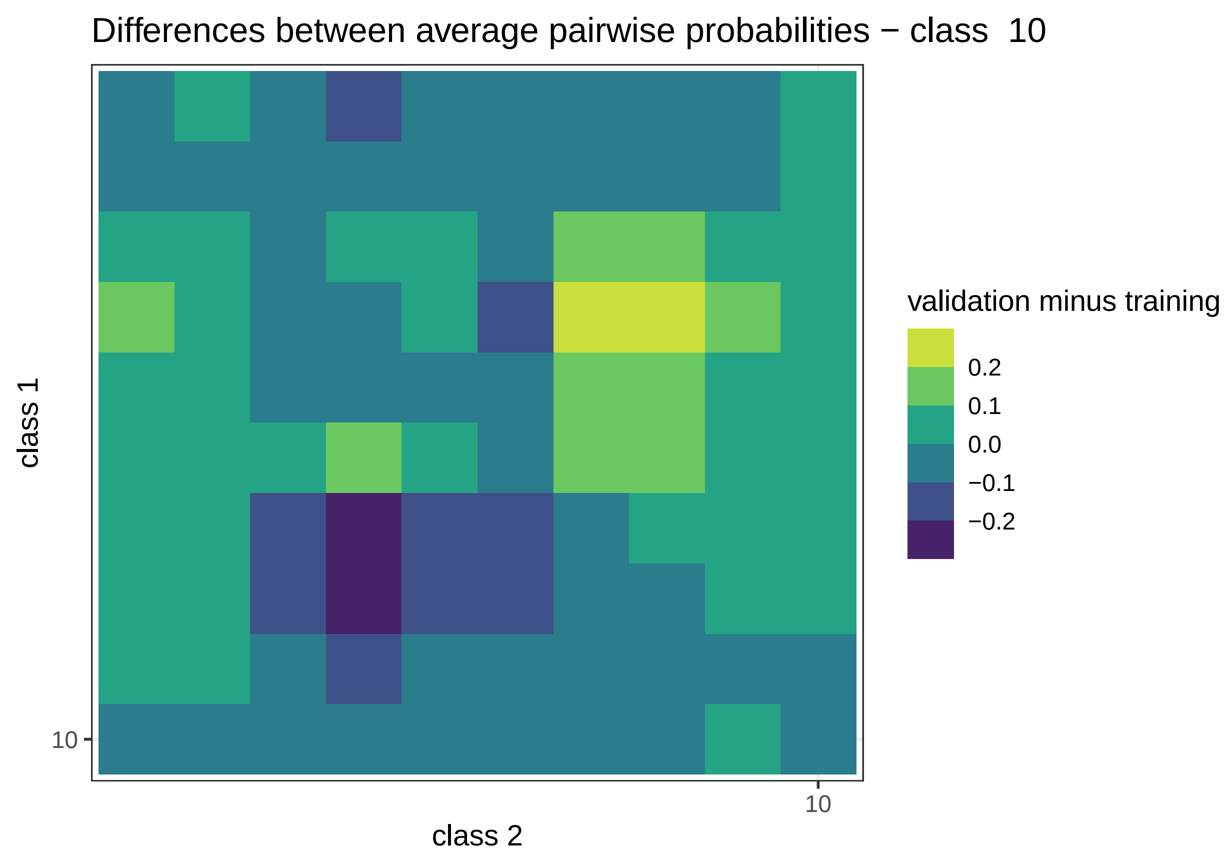

Differences between average pairwise probabilities − class 6
class 1
10
class 2
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2


Differences between average pairwise probabilities − class 7
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2



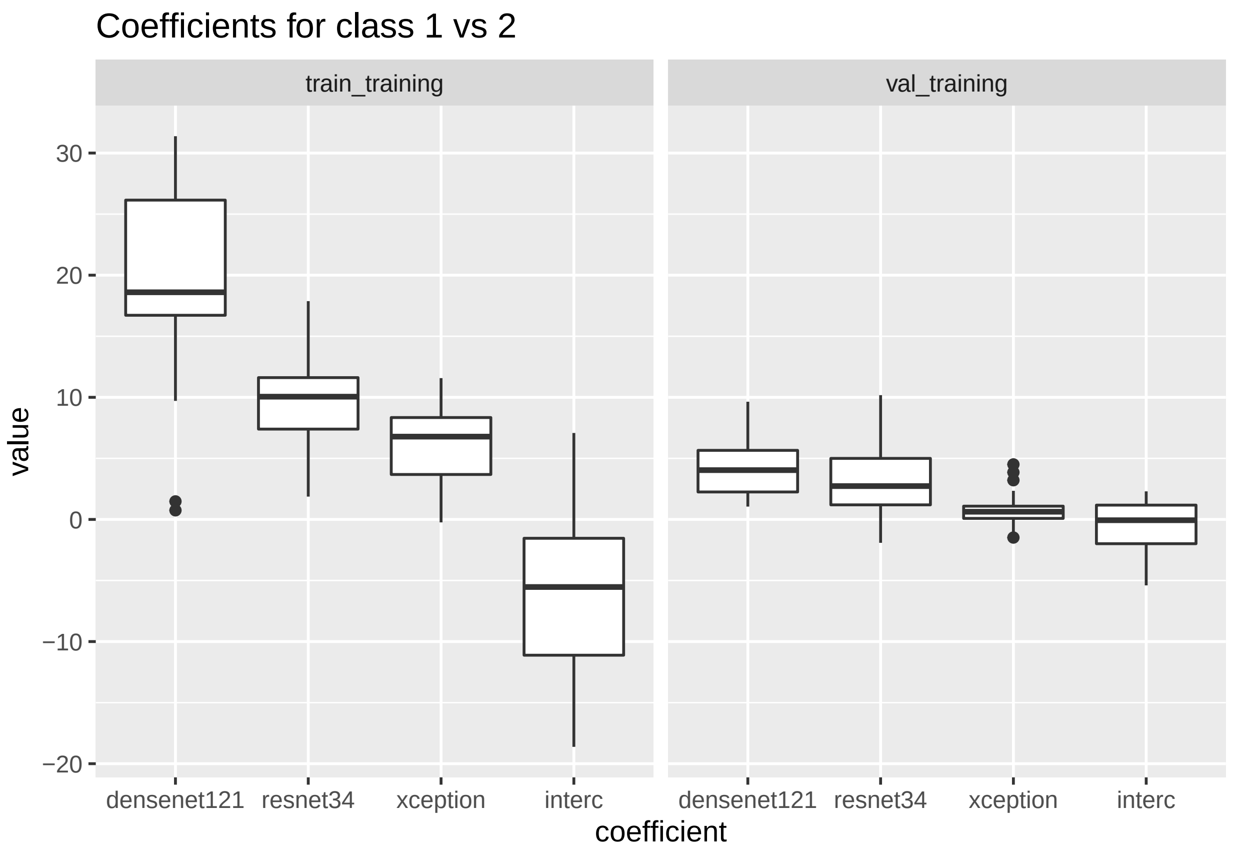
Differences between average pairwise probabilities − class 8
class 1
10
10
class 2
validation minus training
0.2
0.1
0.0
−0.1
−0.2

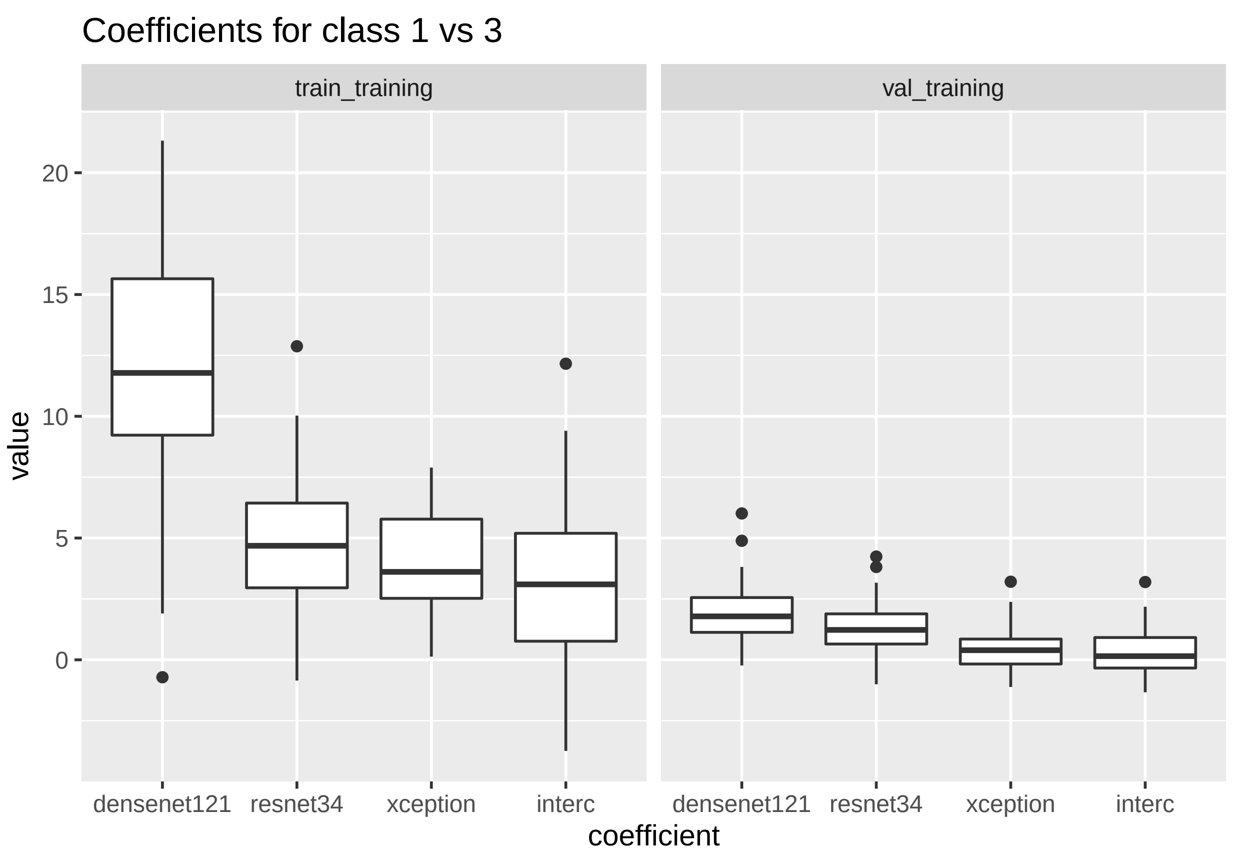

Differences between average pairwise probabilities − class 9
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

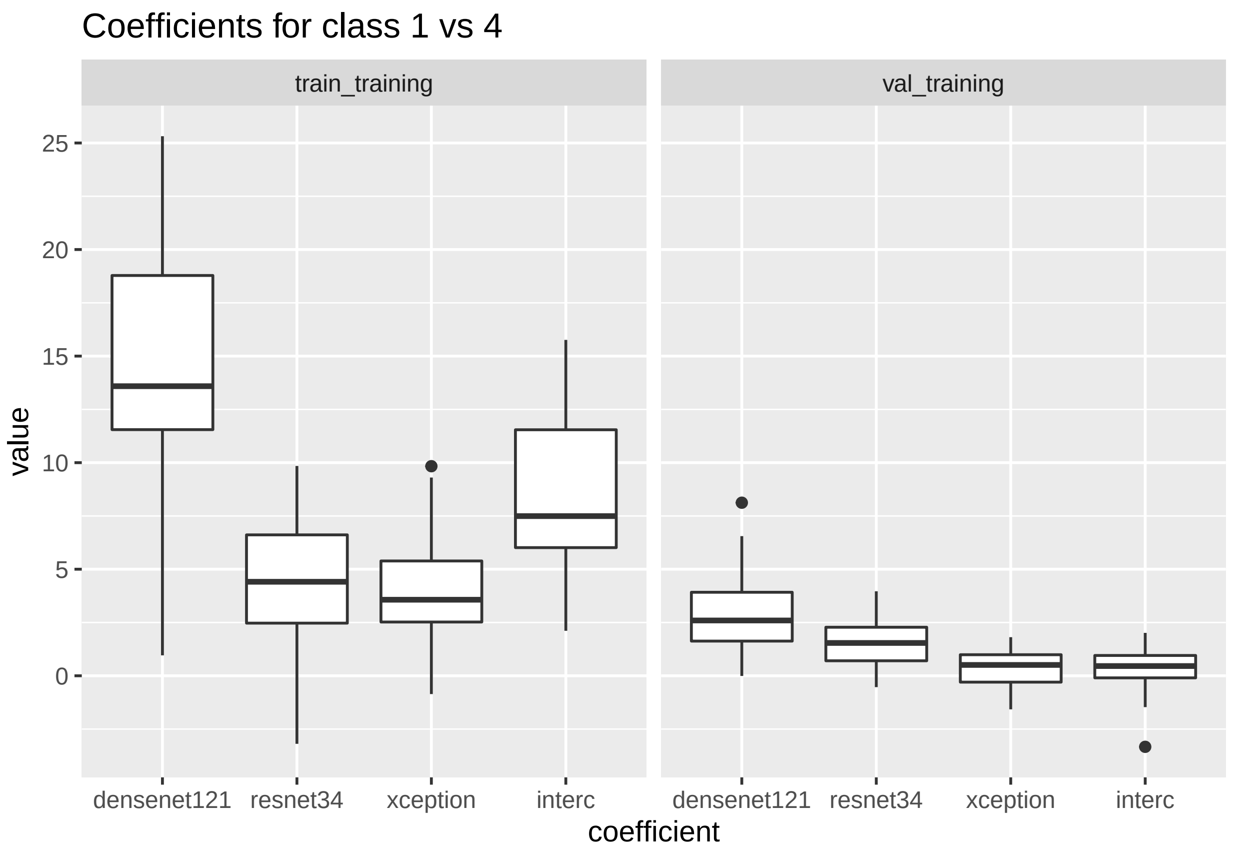


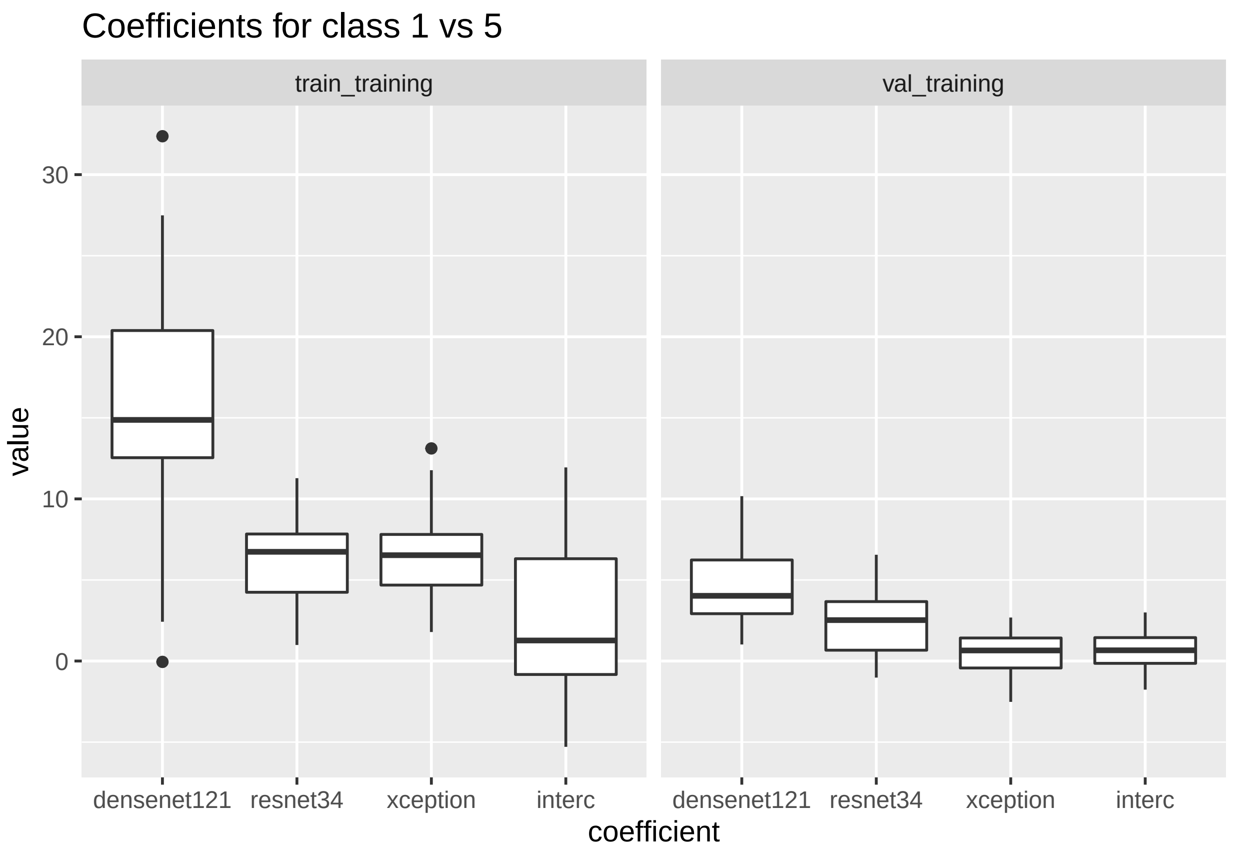
```r
lda_coefs <- load_lda_coefs(base_dir, repls)
```

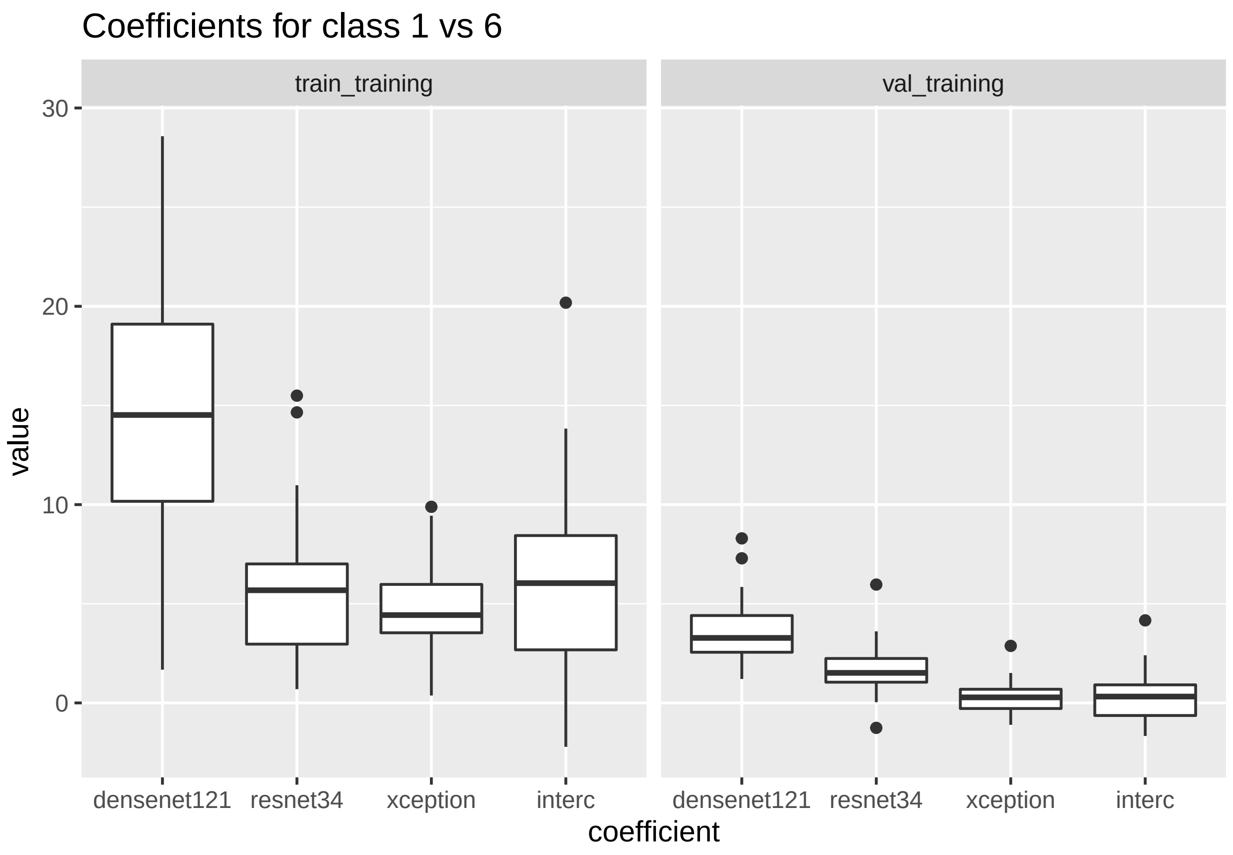
```r
for (cl1 in 1:(classes - 1))
{
  for (cl2 in (cl1 + 1):classes)
  {
    cur_plt <- lda_coefs %>% filter(class1 == cl1 & class2 == cl2) %>% ggplot() + geom_boxplot(aes(x=co
      facet_wrap(~train_type) + ggtitle(paste("Coefficients for class", cl1, "vs", cl2))
    print(cur_plt)
  }
}
```


Coefficients for class 1 vs 2
train_training
val_training
value
30
20
10
0
−10
−20
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

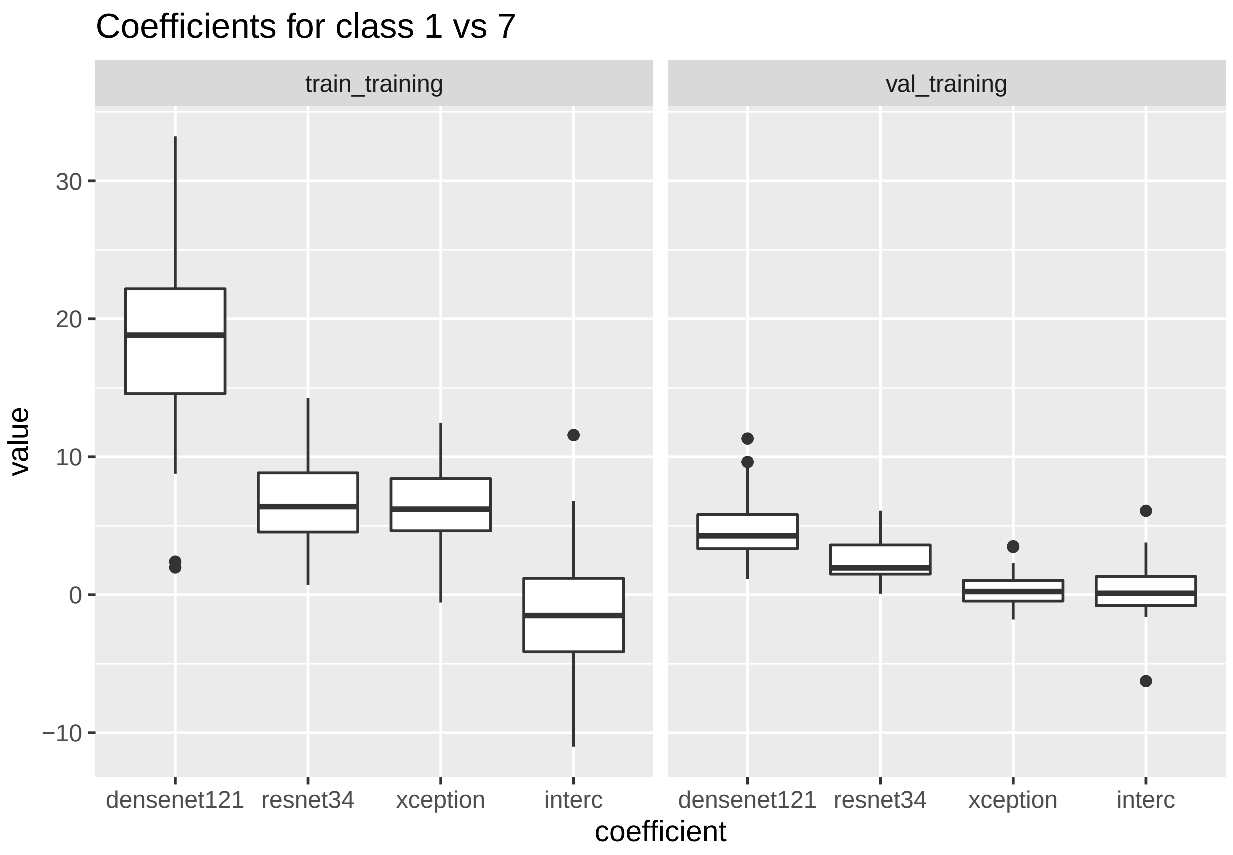

Coefficients for class 1 vs 3
train_training
val_training
value
coefficient
densenet121
resnet34
xception
interc
0
5
10
15
20

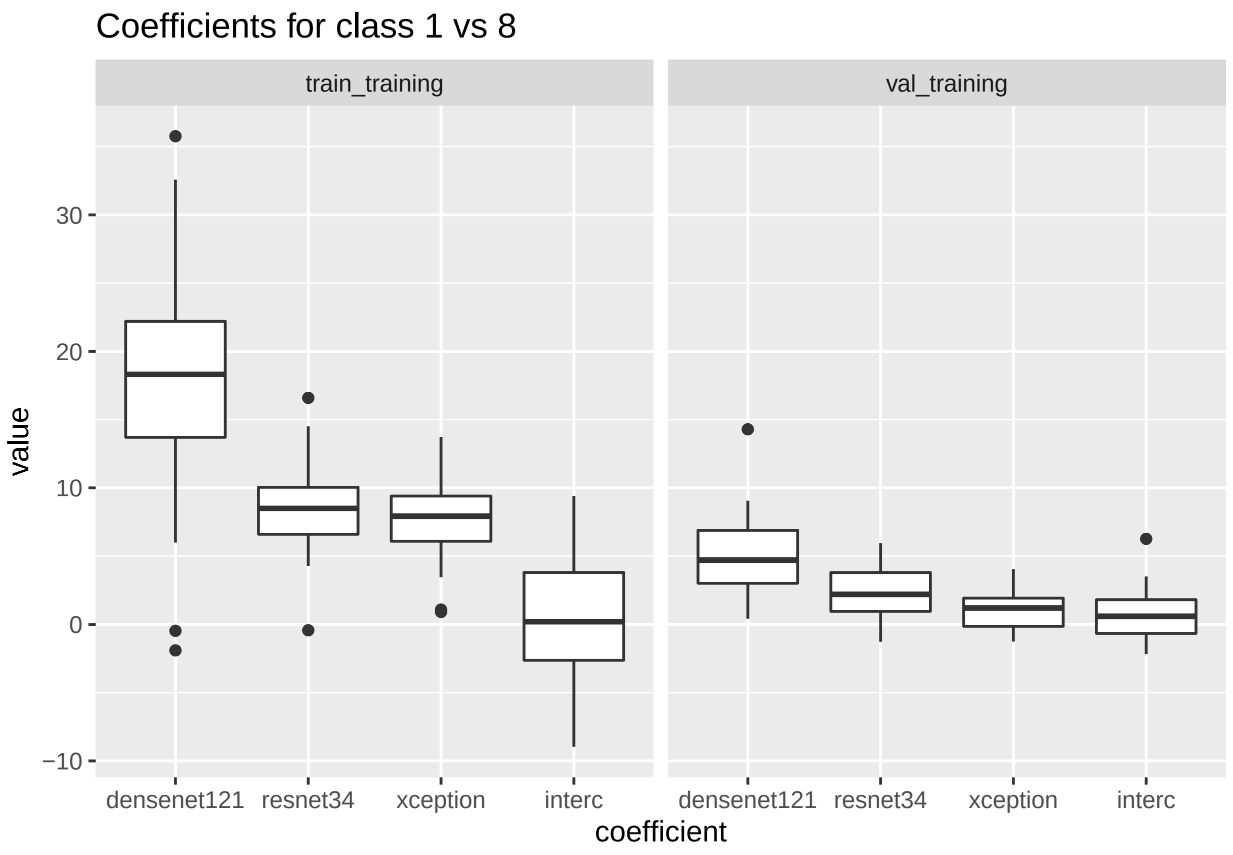

Coefficients for class 1 vs 4
train_training
val_training
value
25
20
15
10
5
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

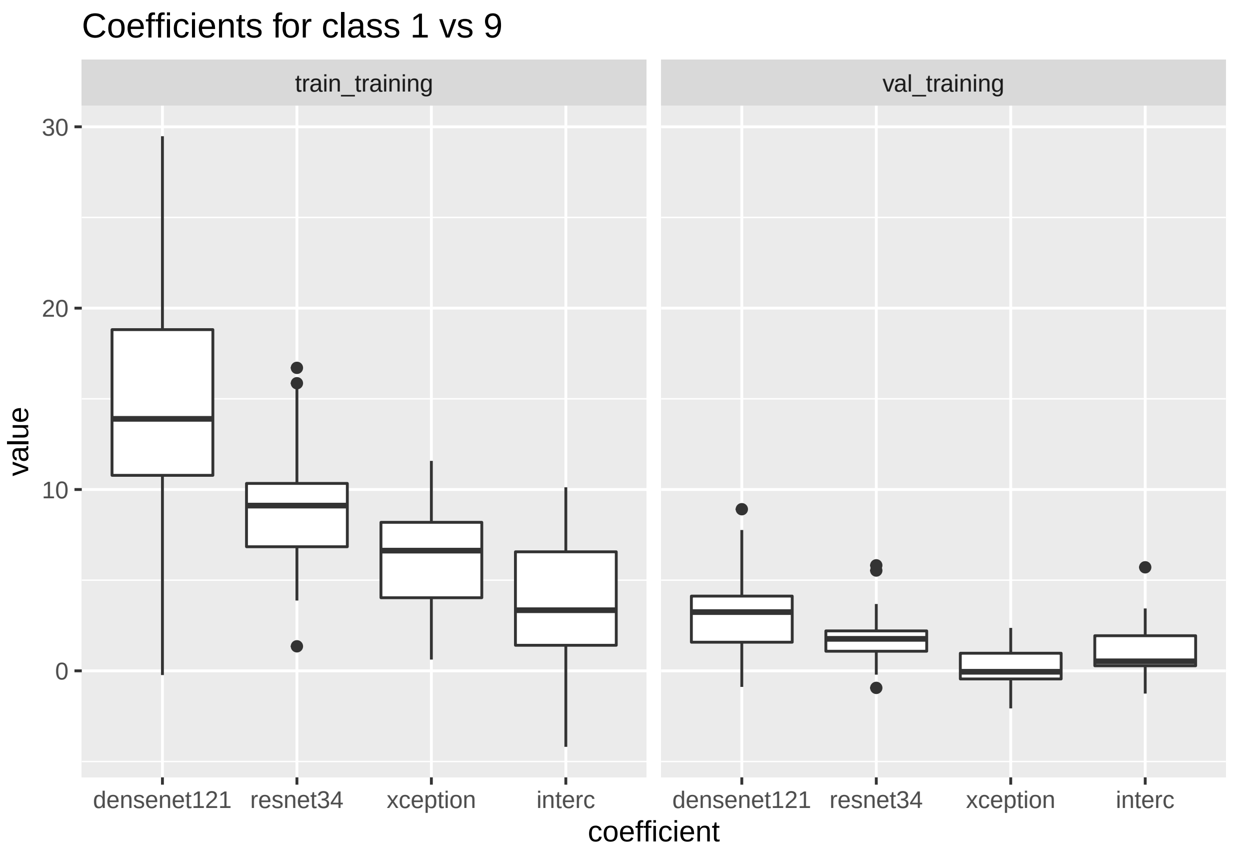

Coefficients for class 1 vs 5
train_training
val_training
value
30
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

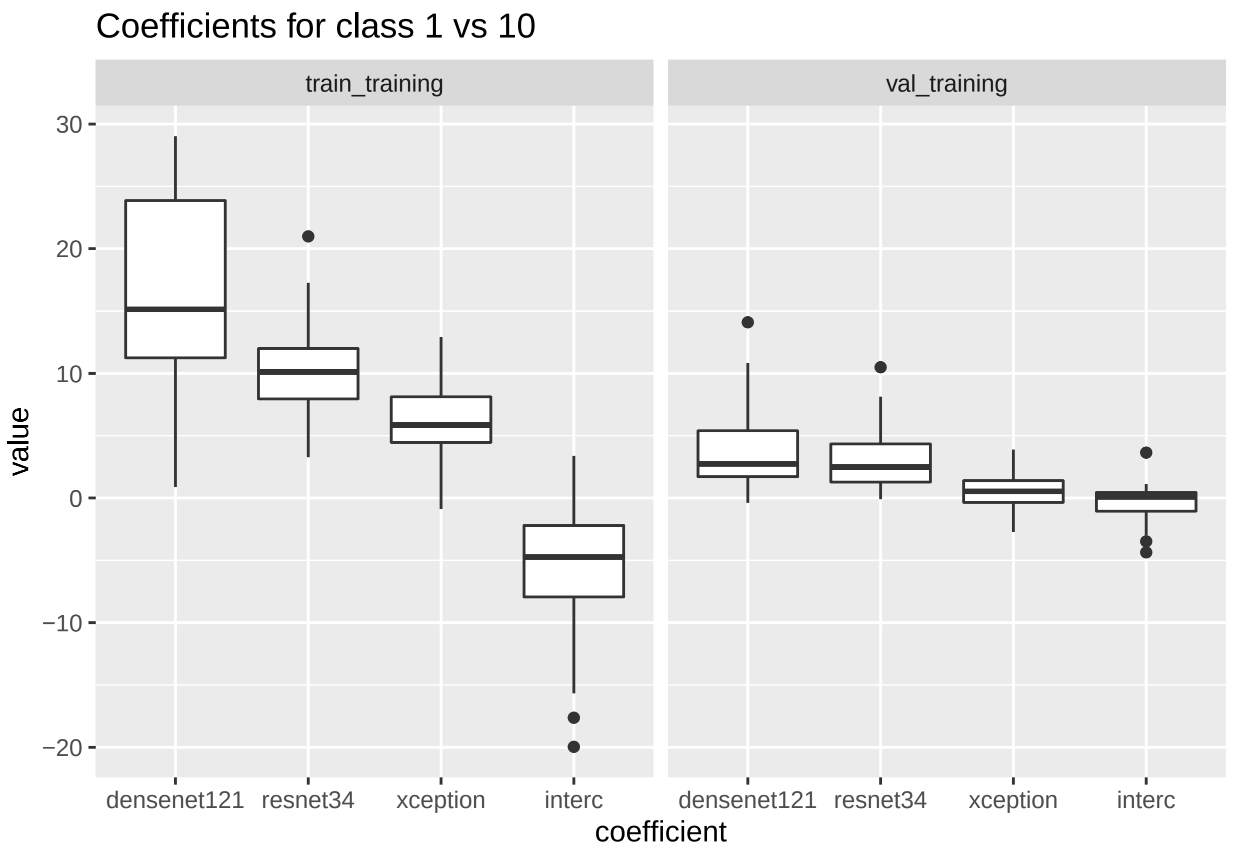


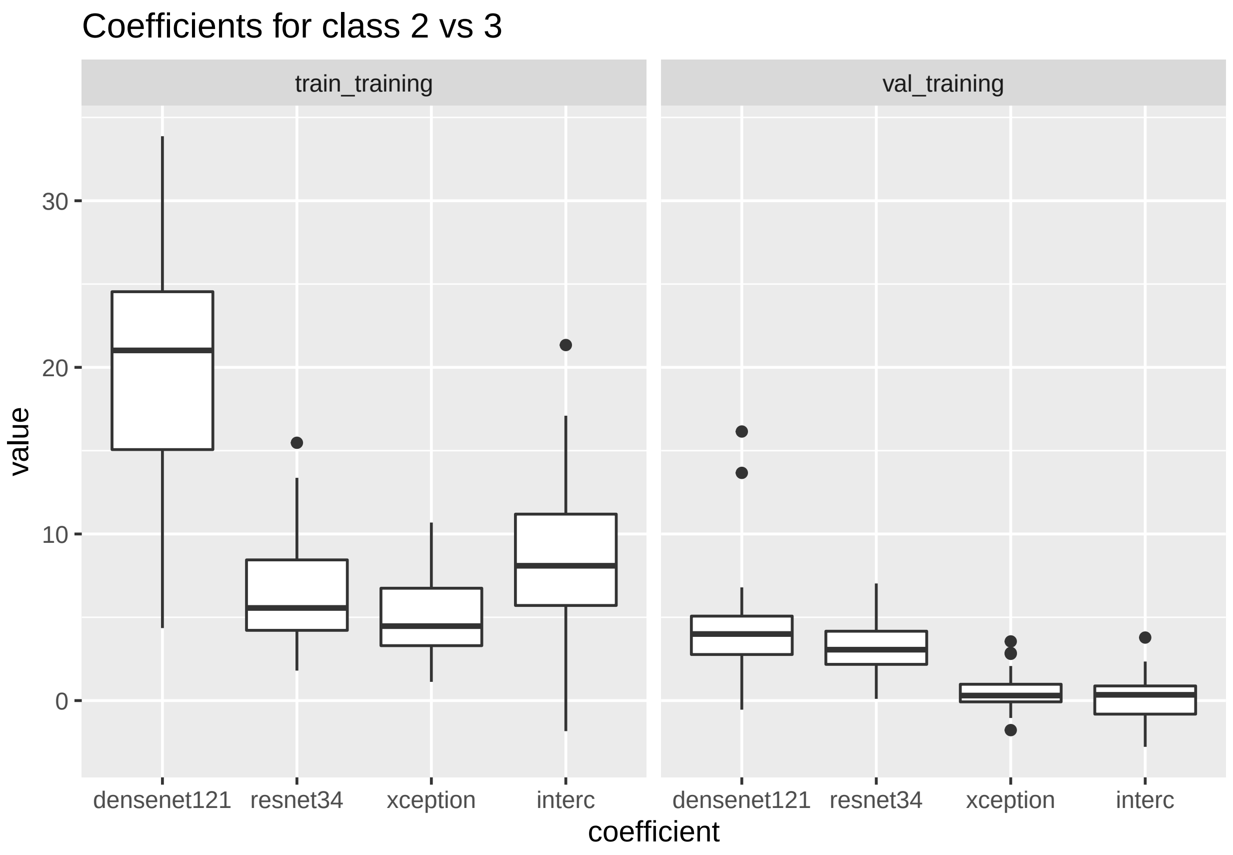
Coefficients for class 1 vs 6
train_training
val_training
value
30
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 1 vs 7
train_training
val_training
value
30
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

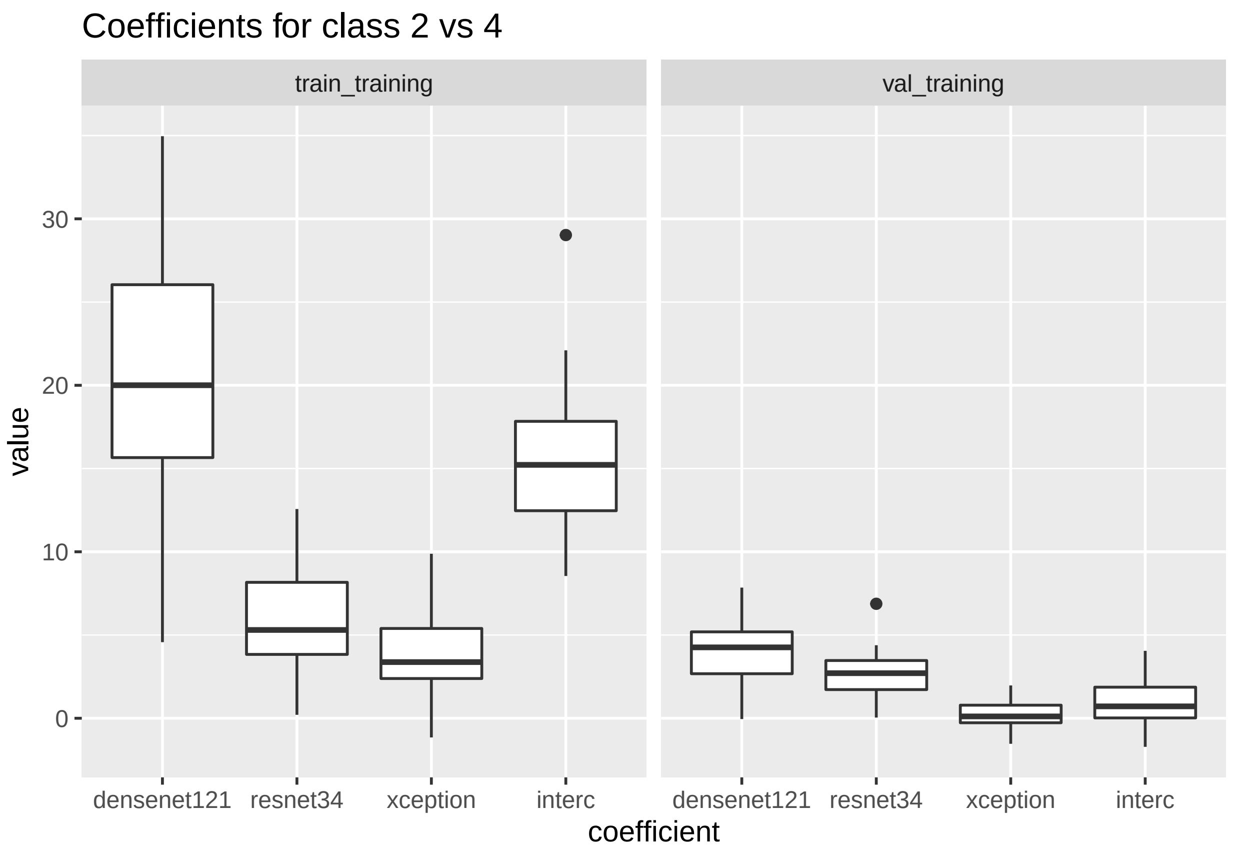

Coefficients for class 1 vs 8
train_training
val_training
value
30
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

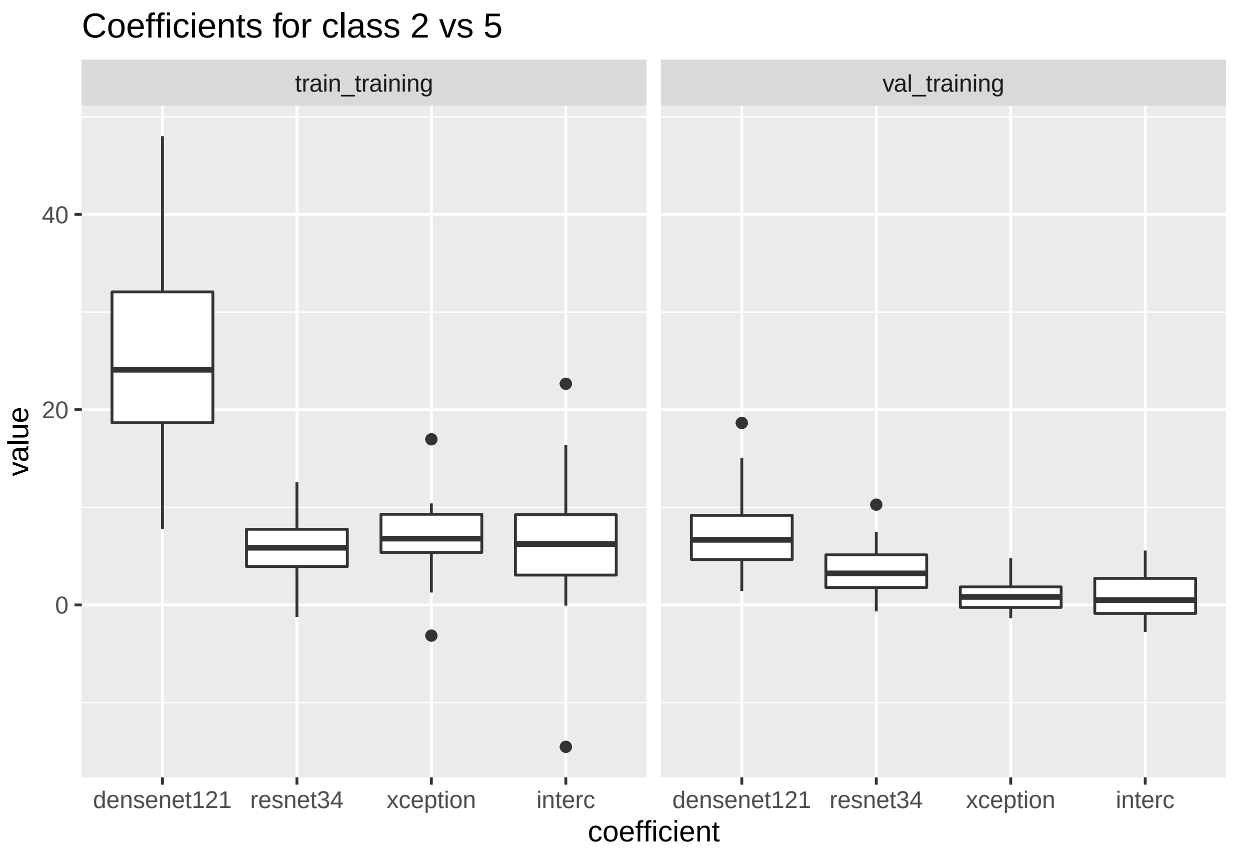

Coefficients for class 1 vs 9
train_training
val_training
value
30
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

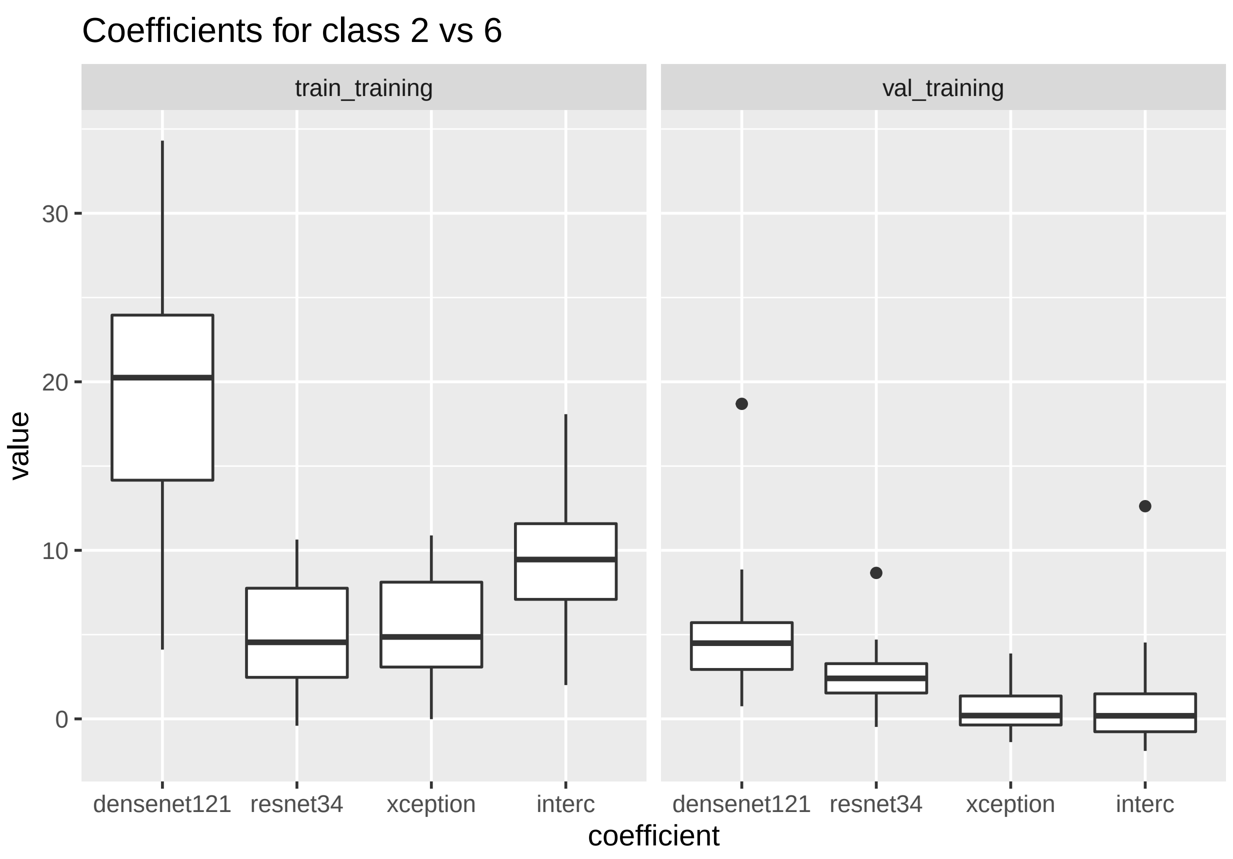

Coefficients for class 1 vs 10
train_training
val_training
value
30
20
10
0
−10
−20
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

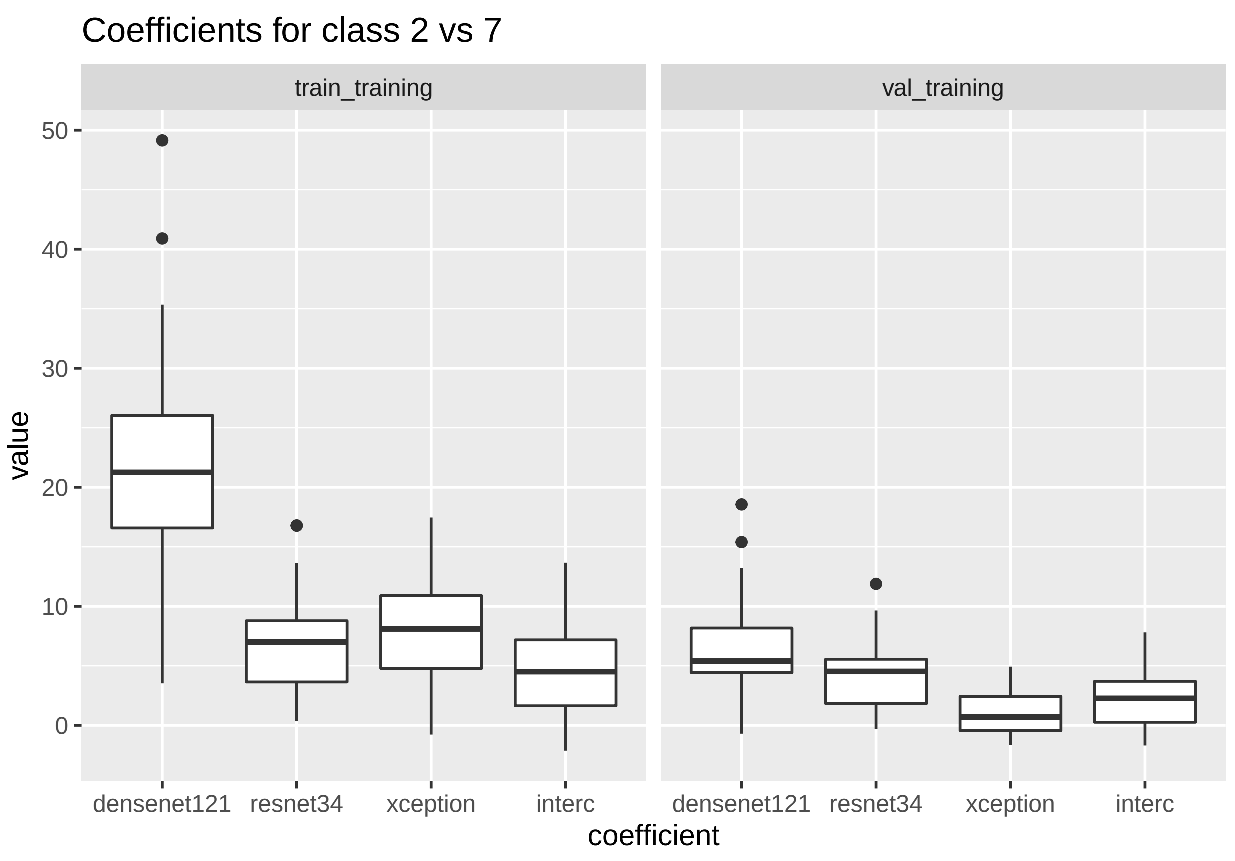


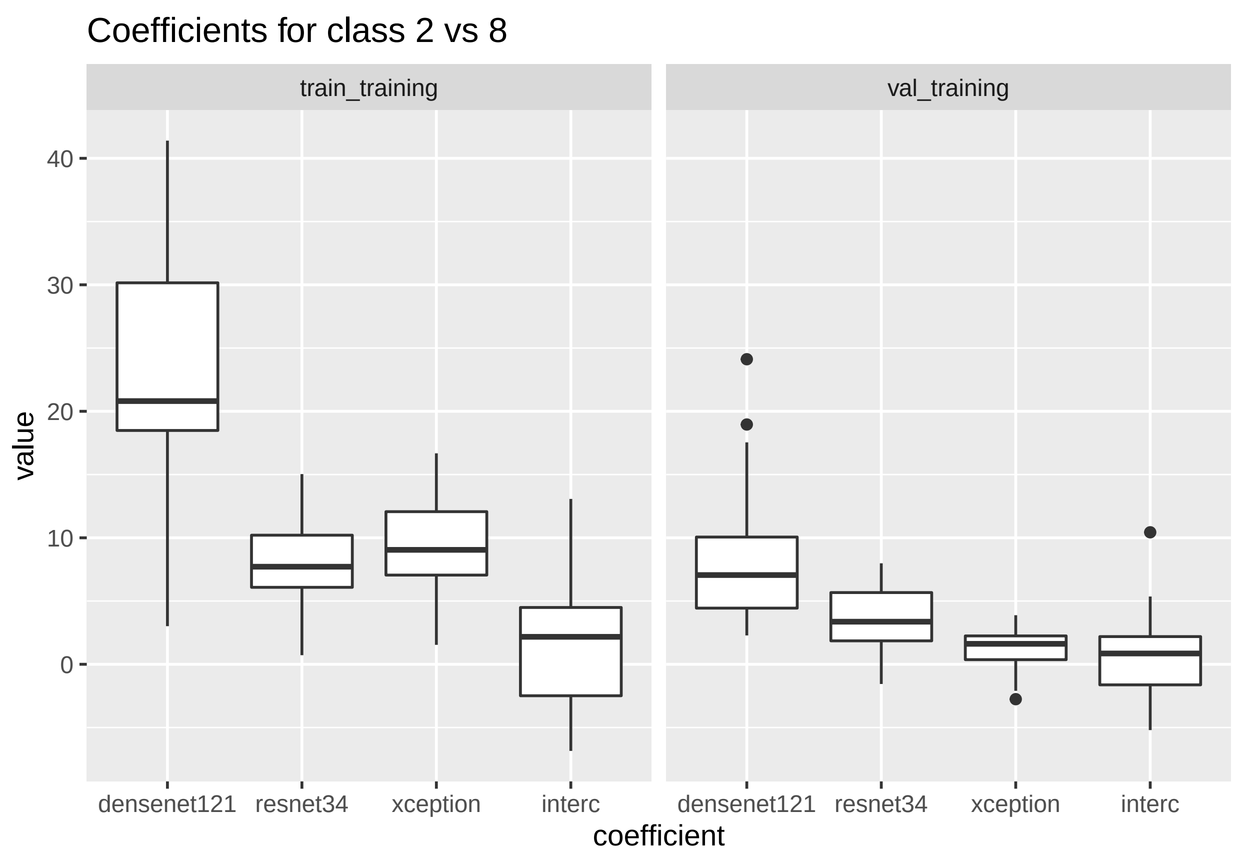
Coefficients for class 2 vs 3
train_training
val_training
value
30
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

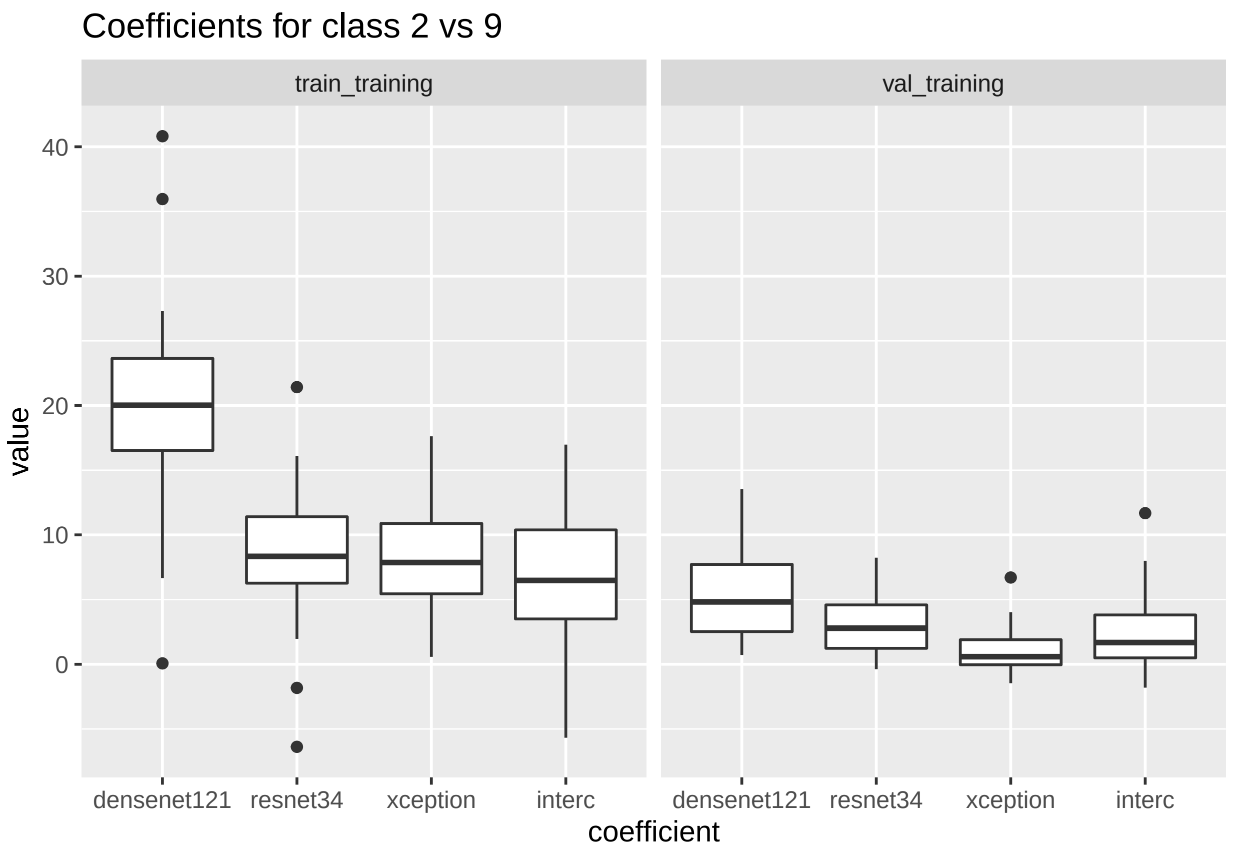

Coefficients for class 2 vs 4
train_training
val_training
value
30
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 2 vs 5
train_training
val_training
value
40
20
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

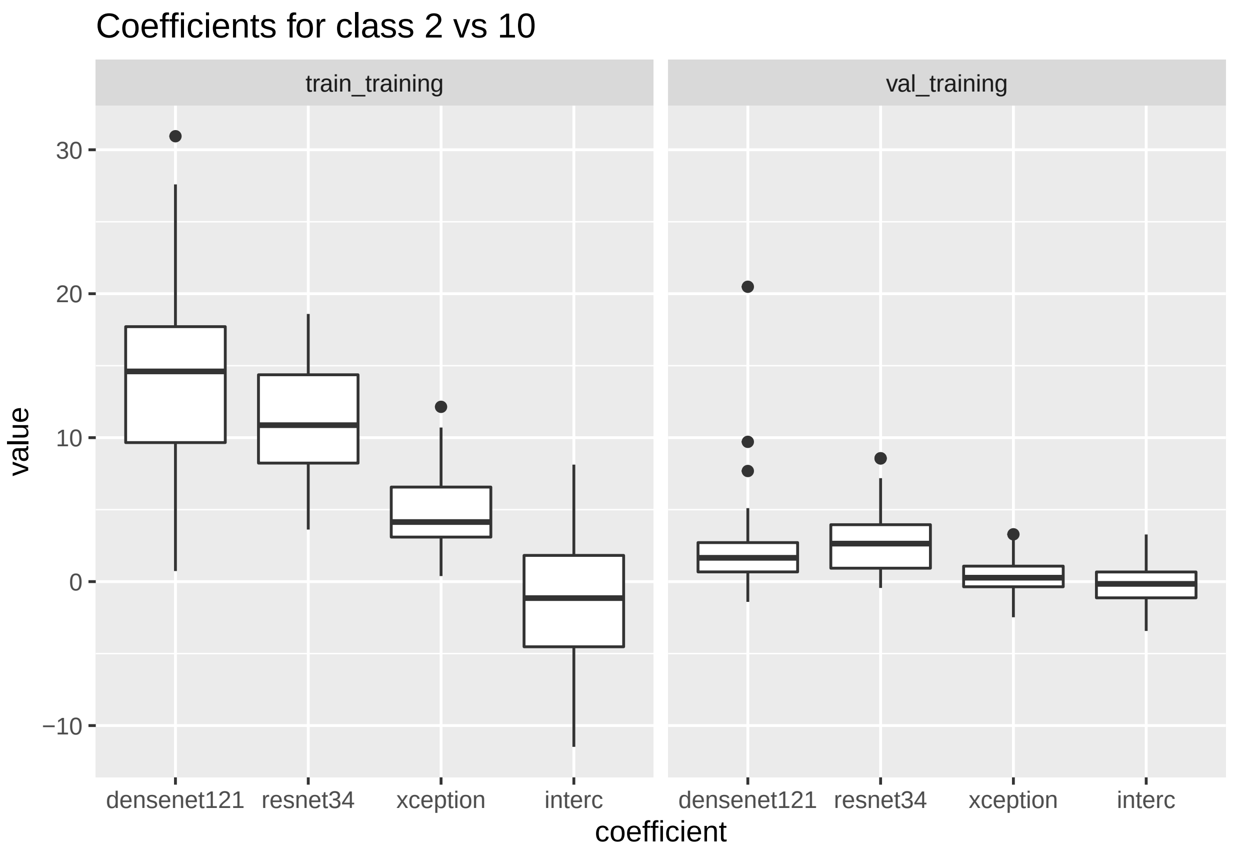

Coefficients for class 2 vs 6
train_training
val_training
value
30
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

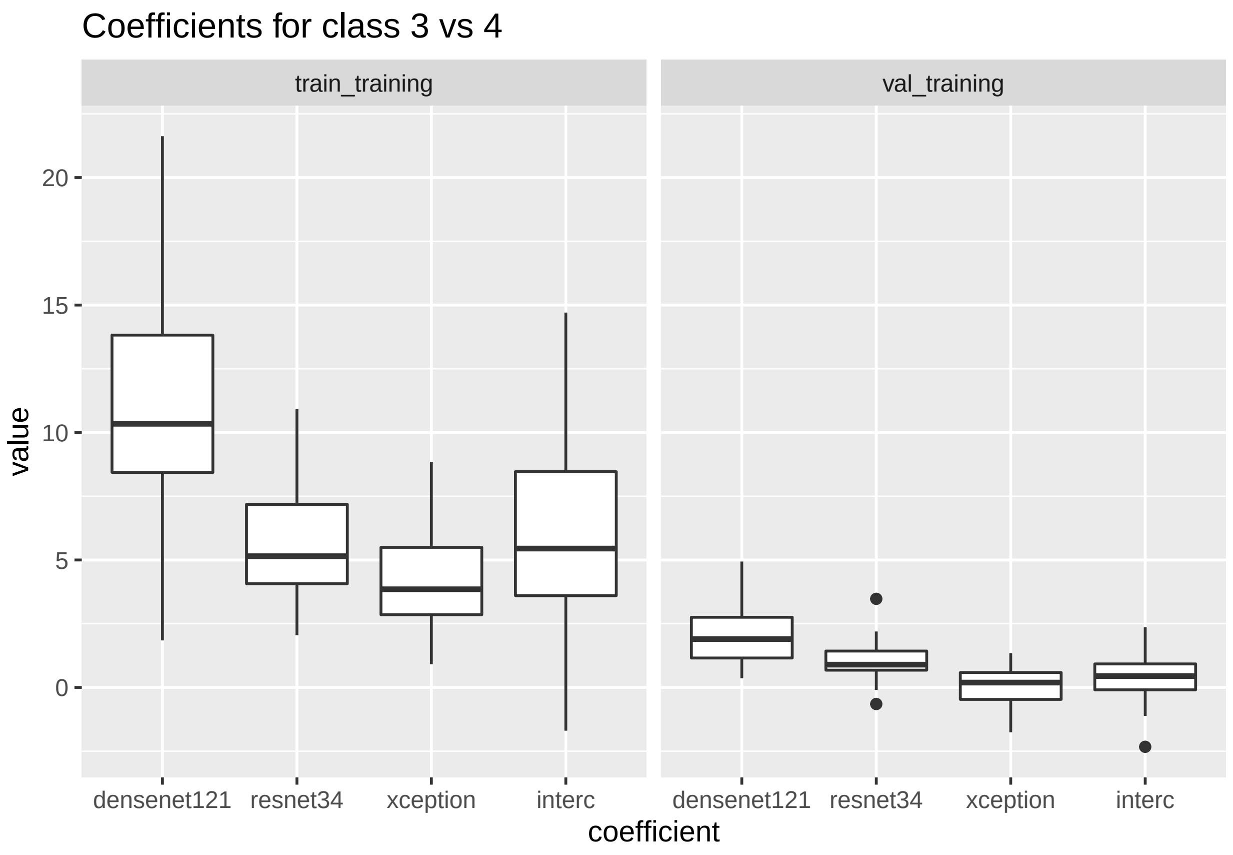

Coefficients for class 2 vs 7
train_training
val_training
value
coefficient
densenet121
resnet34
xception
interc
0
10
20
30
40
50


Coefficients for class 2 vs 8
train_training
val_training
value
40
30
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

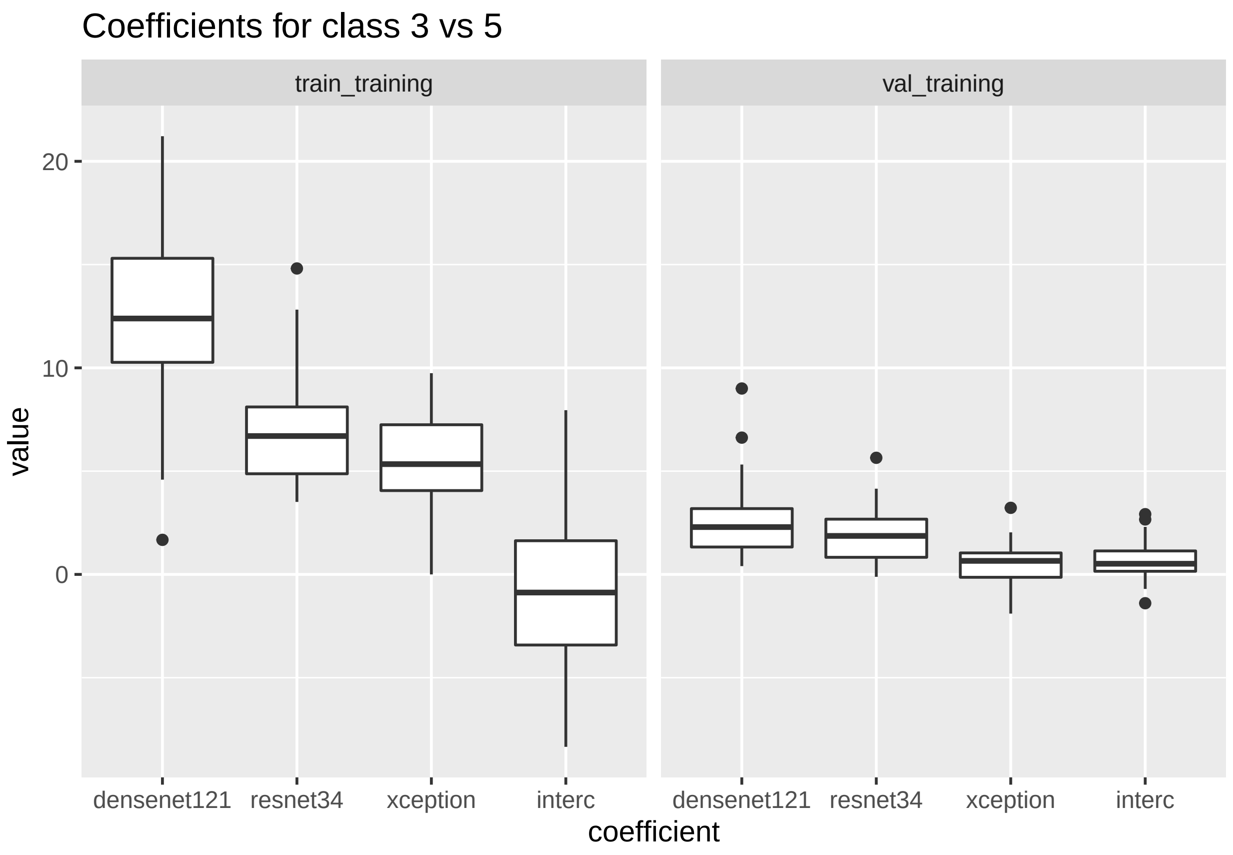


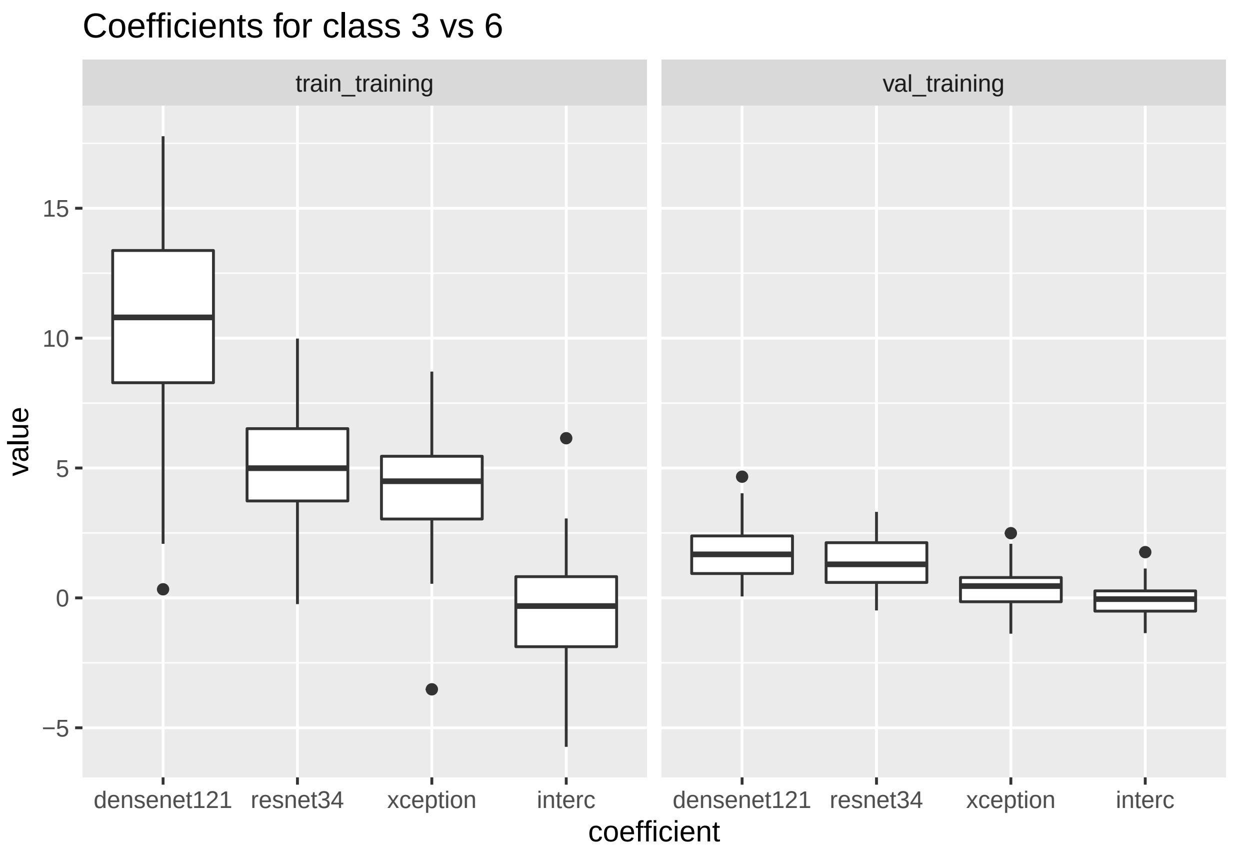
Coefficients for class 2 vs 9
train_training
val_training
value
40
30
20
10
0
densenet121
resnet34
xception
interc
coefficient

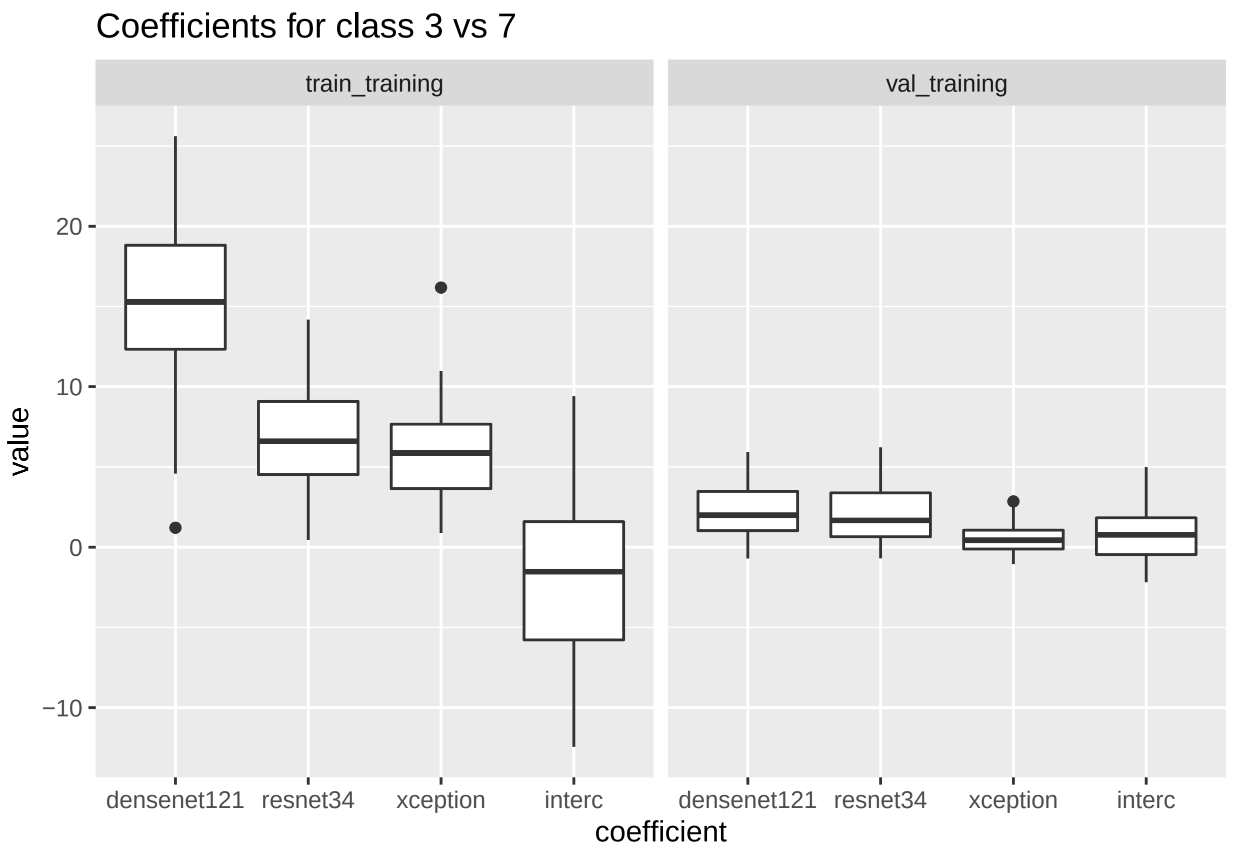

Coefficients for class 2 vs 10
train_training
val_training
value
30
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

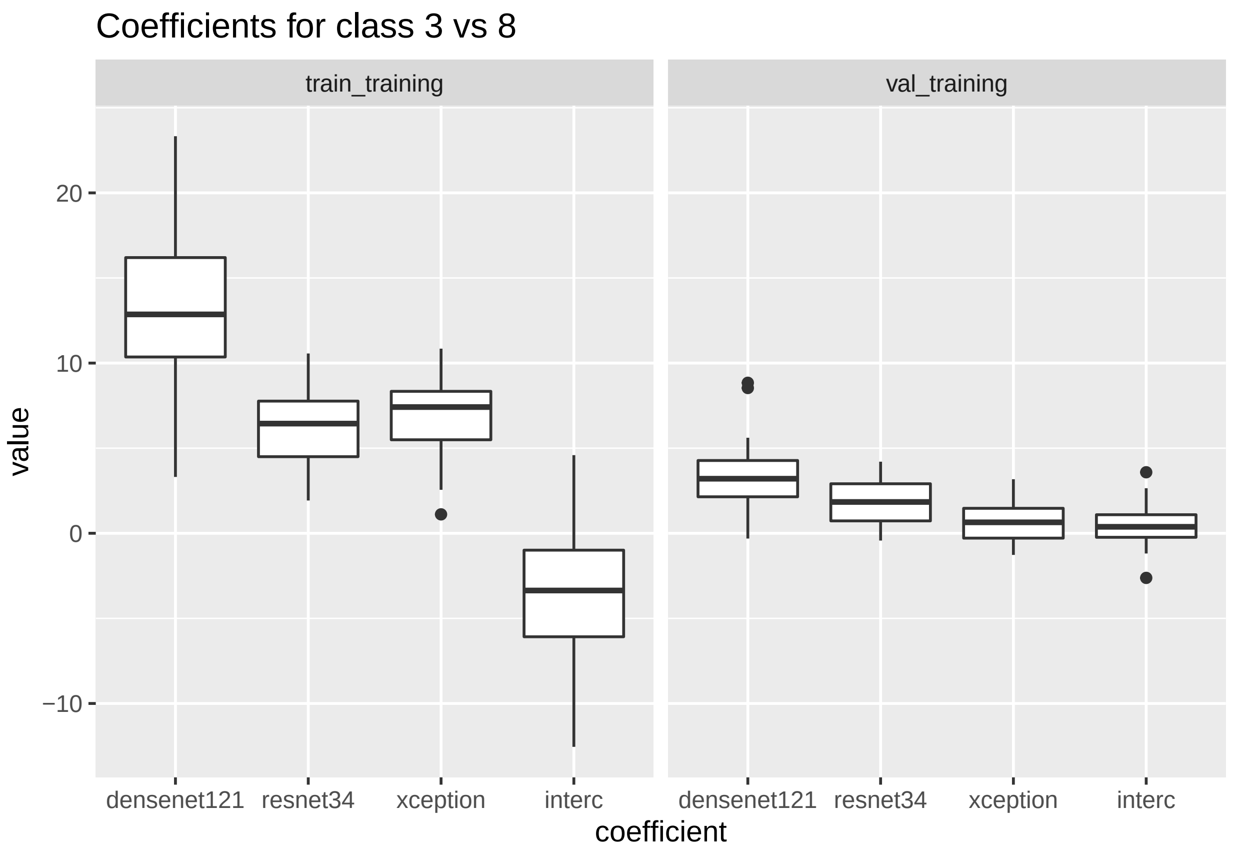


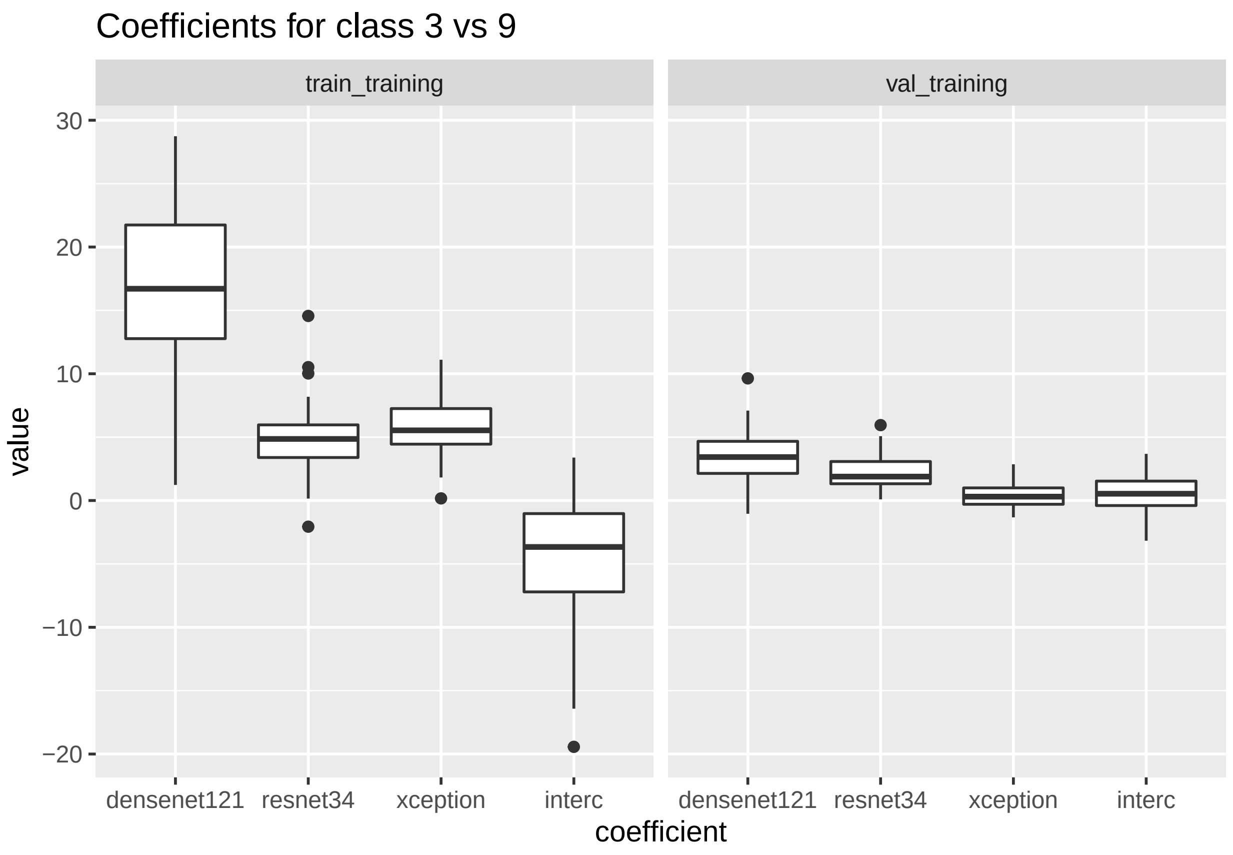
Coefficients for class 3 vs 4
train_training
val_training
value
20
15
10
5
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

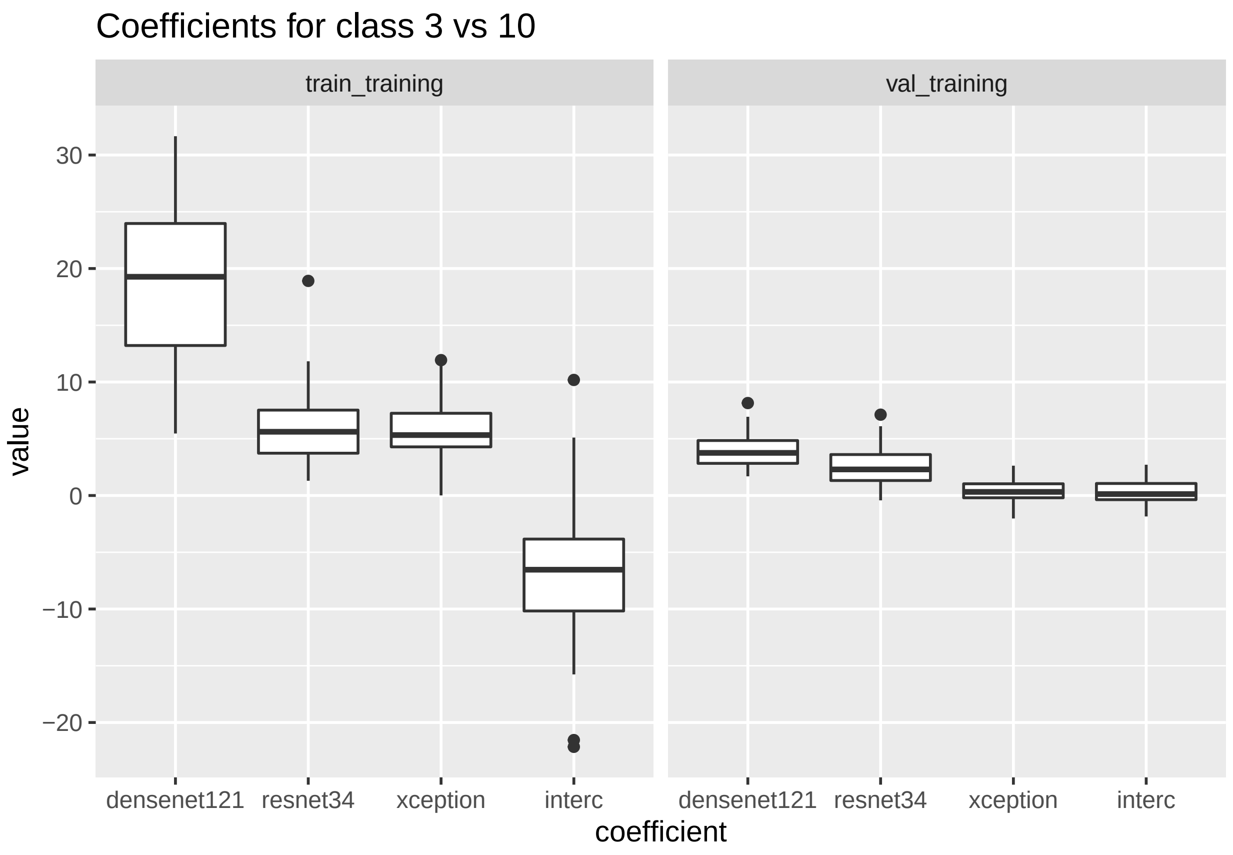

Coefficients for class 3 vs 5
train_training
val_training
value
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

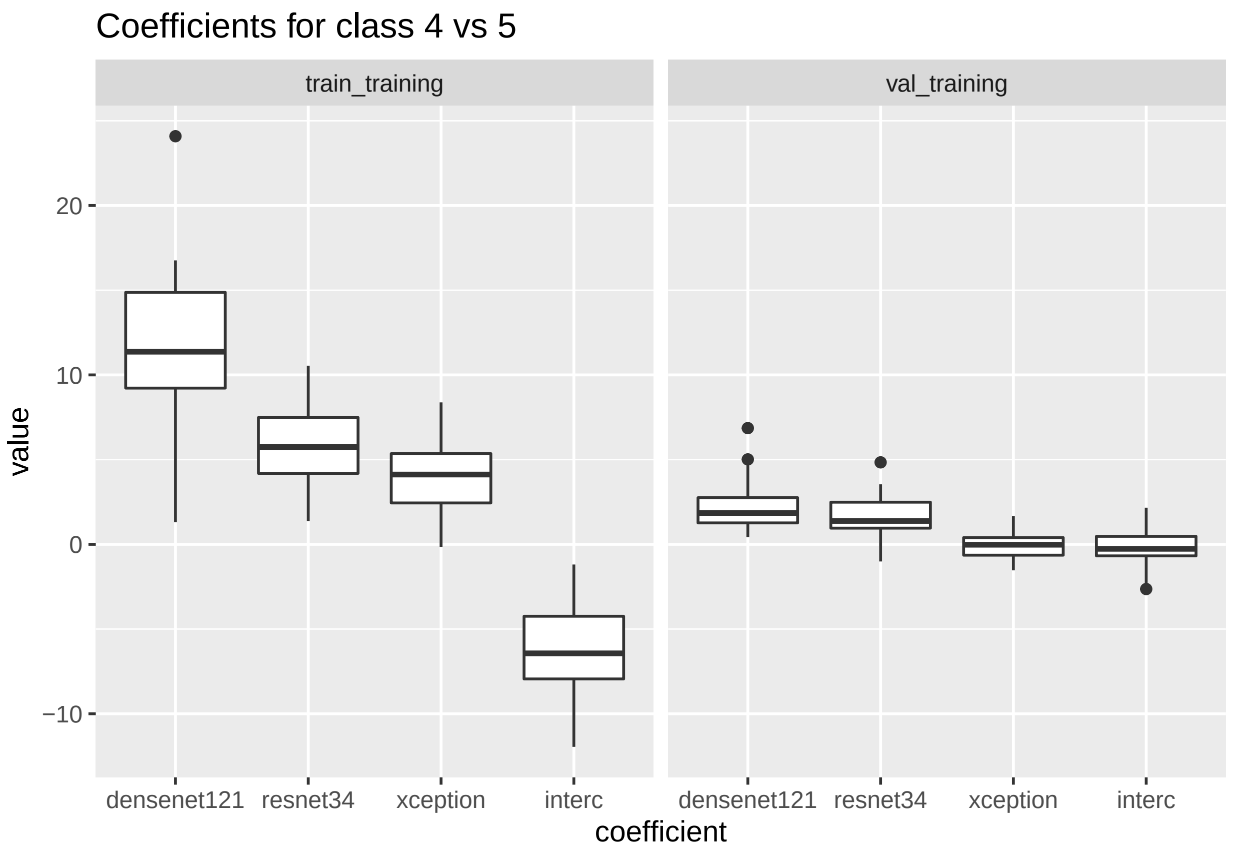

Coefficients for class 3 vs 6
train_training
val_training
value
15
10
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 3 vs 7
train_training
val_training
value
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

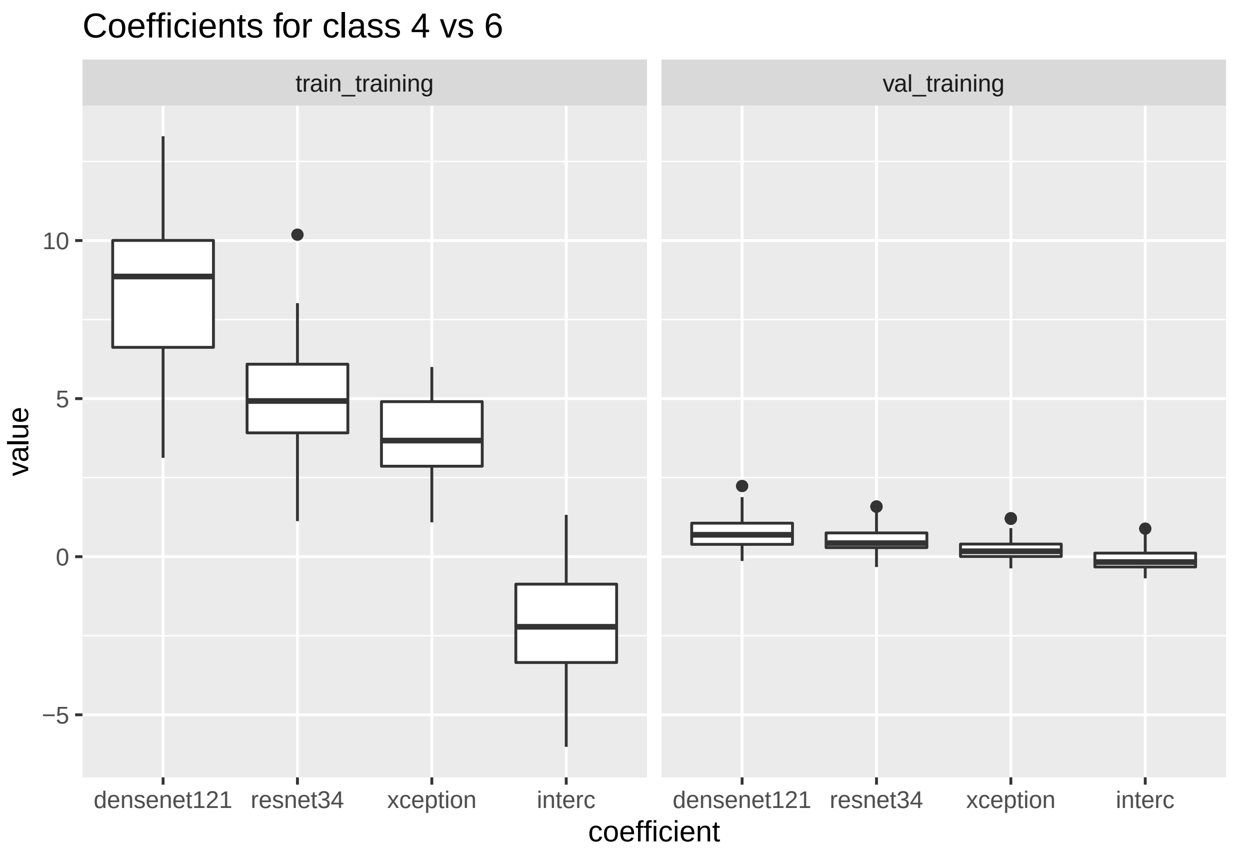

Coefficients for class 3 vs 8
train_training
val_training
value
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

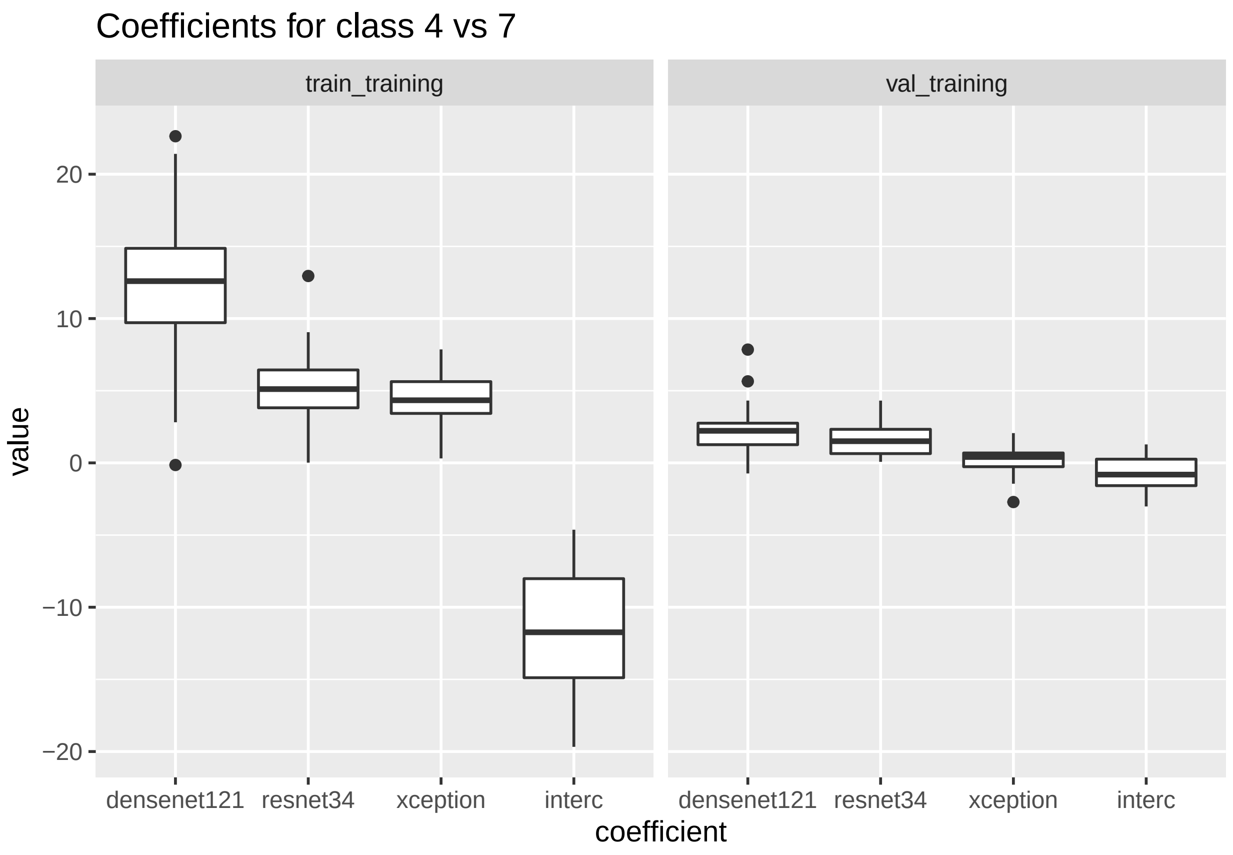

Coefficients for class 3 vs 9
train_training
val_training
30
20
10
0
−10
−20
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

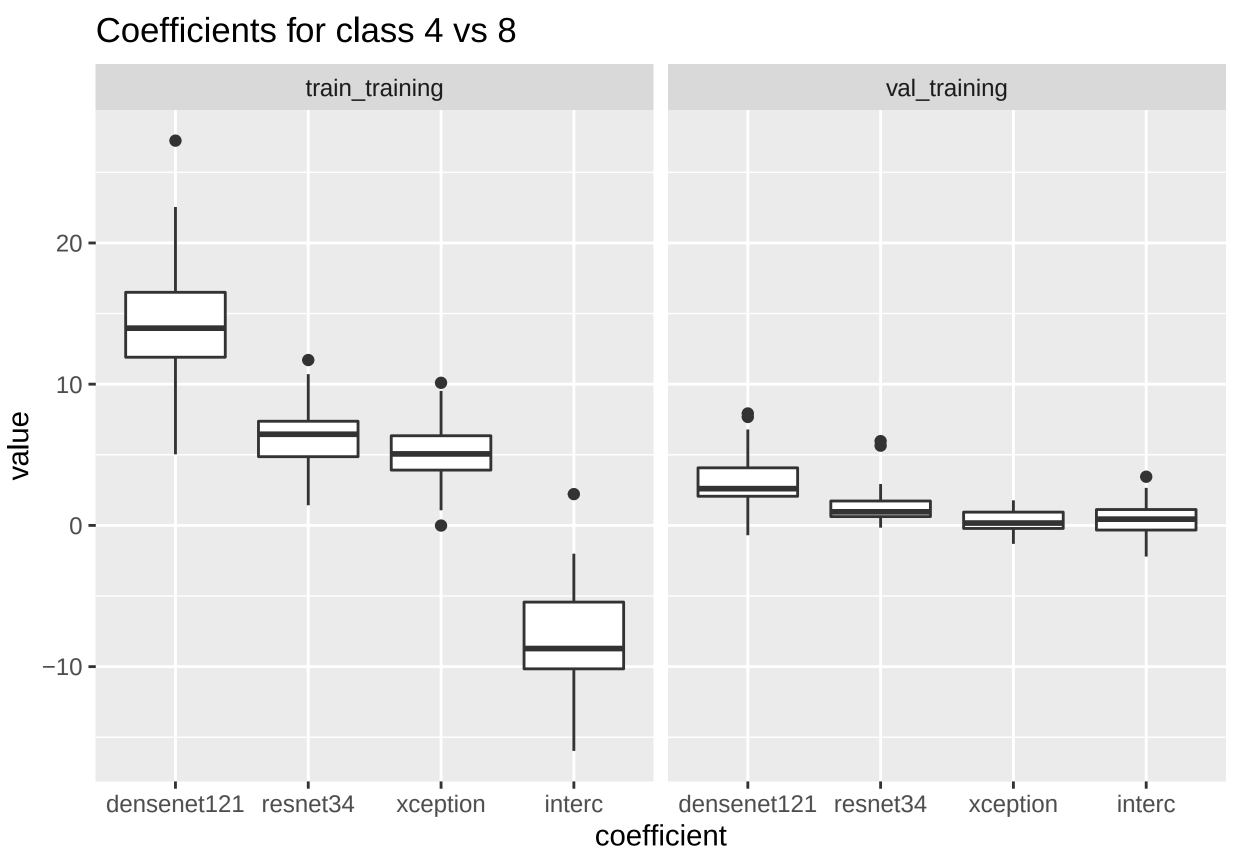

Coefficients for class 3 vs 10
train_training
val_training
value
30
20
10
0
−10
−20
densenet121
resnet34
xception
interc
coefficient

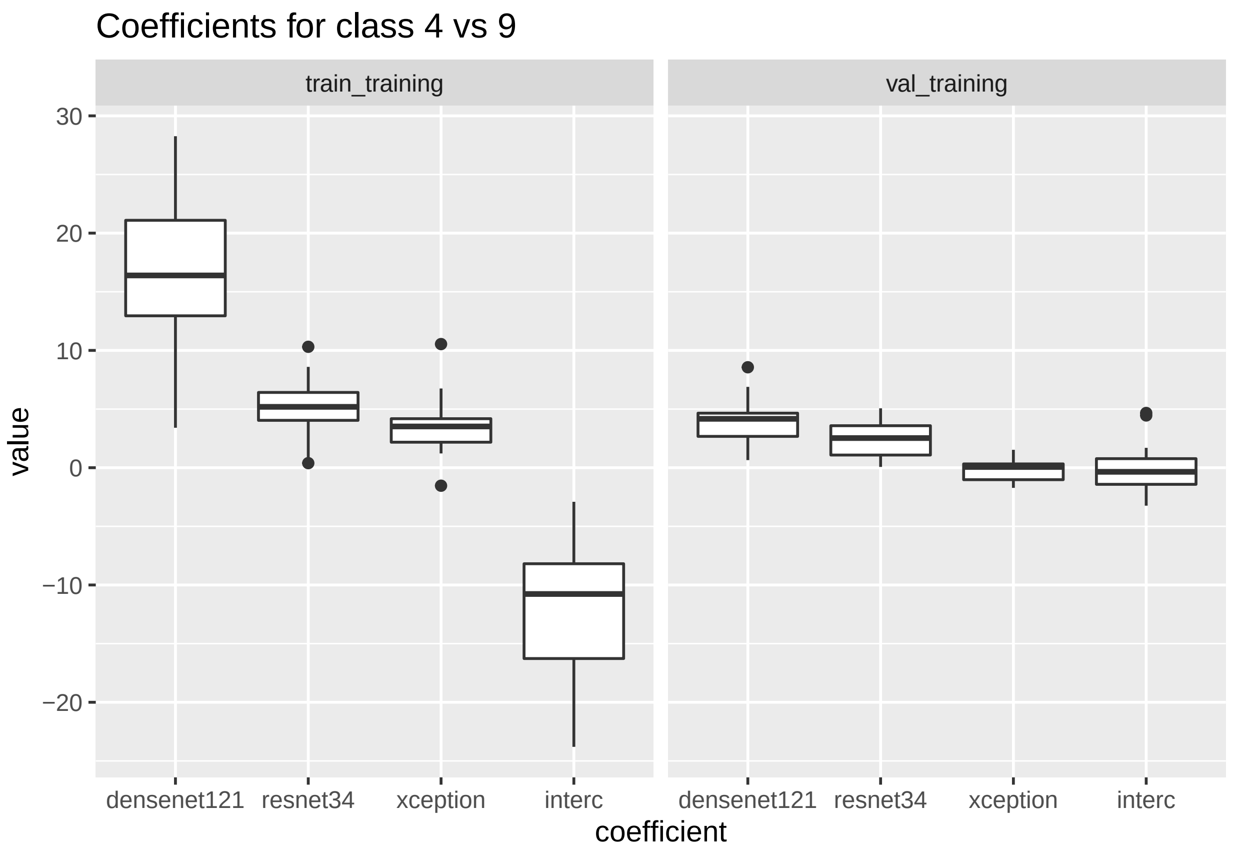


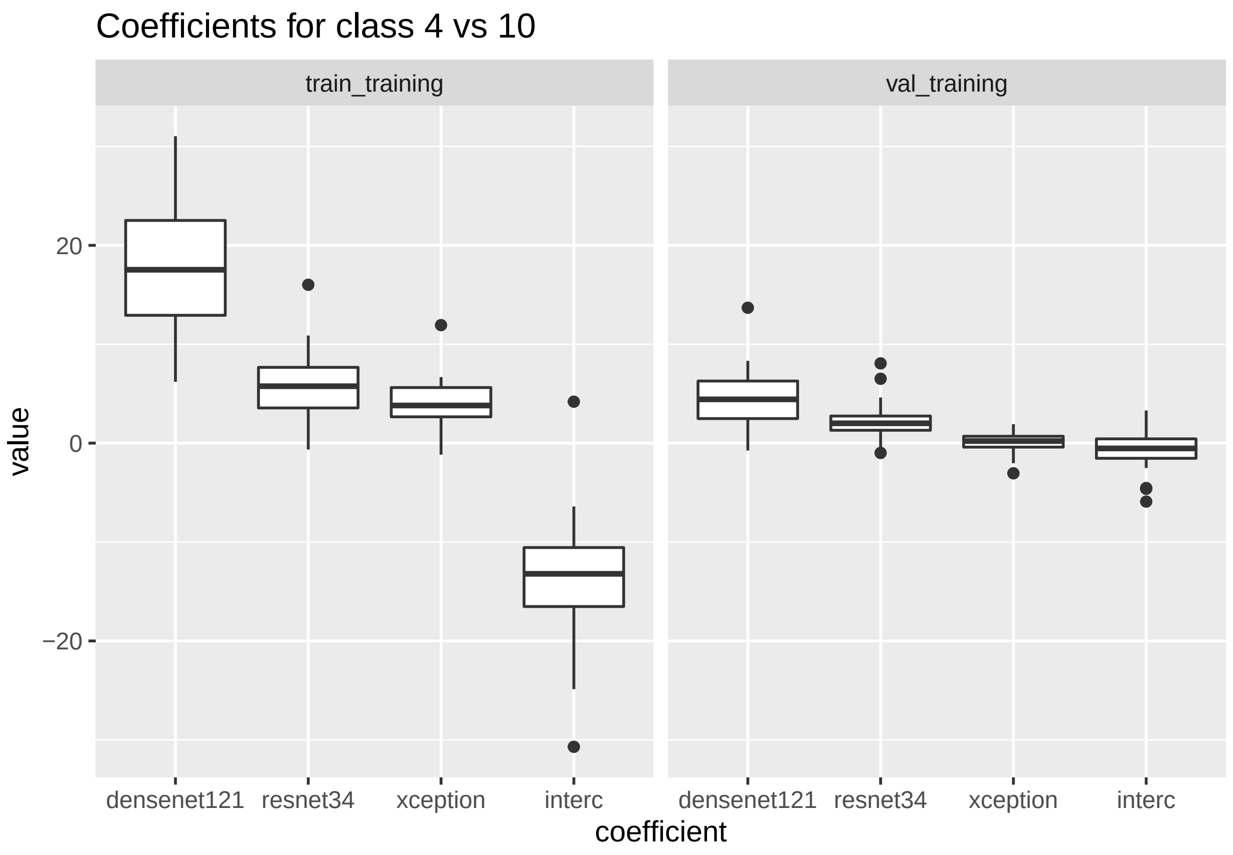
Coefficients for class 4 vs 5
train_training
val_training
value
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

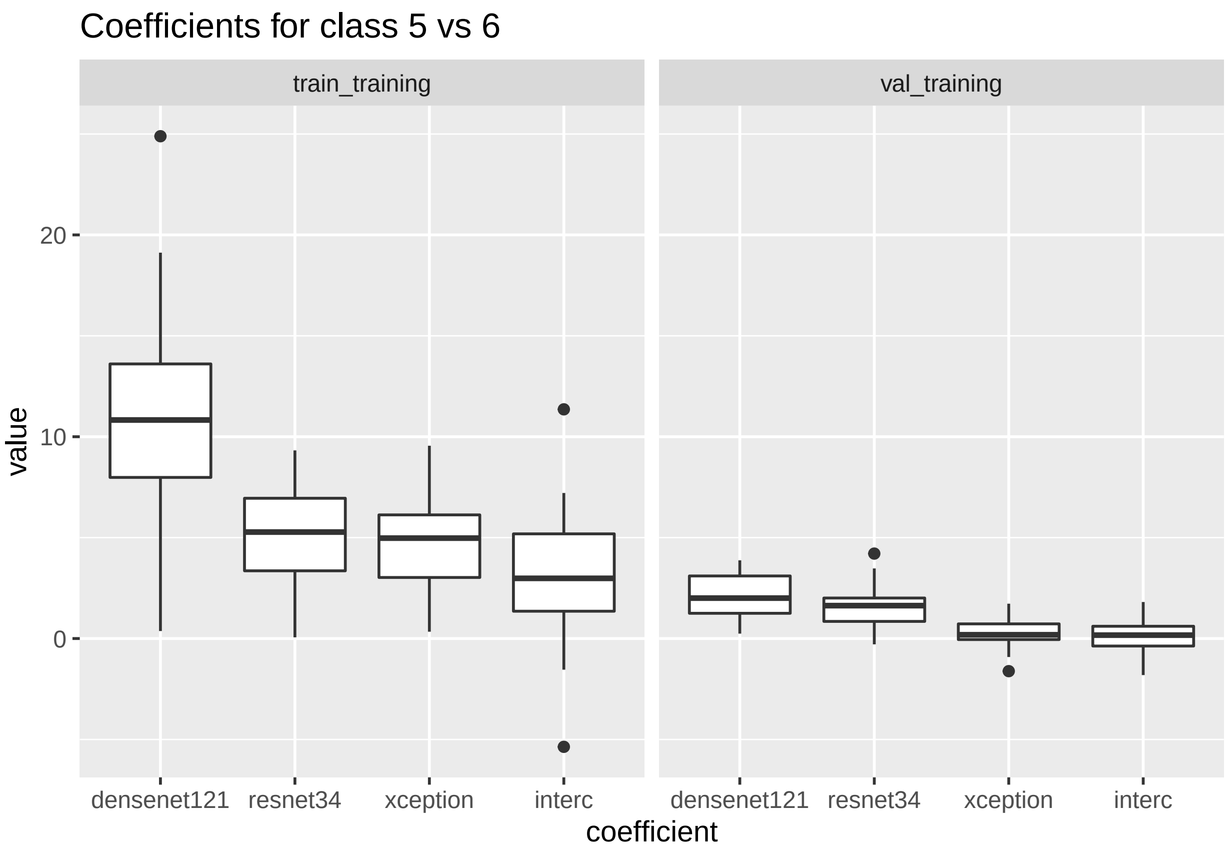

Coefficients for class 4 vs 6
train_training
val_training
value
coefficient
10
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc

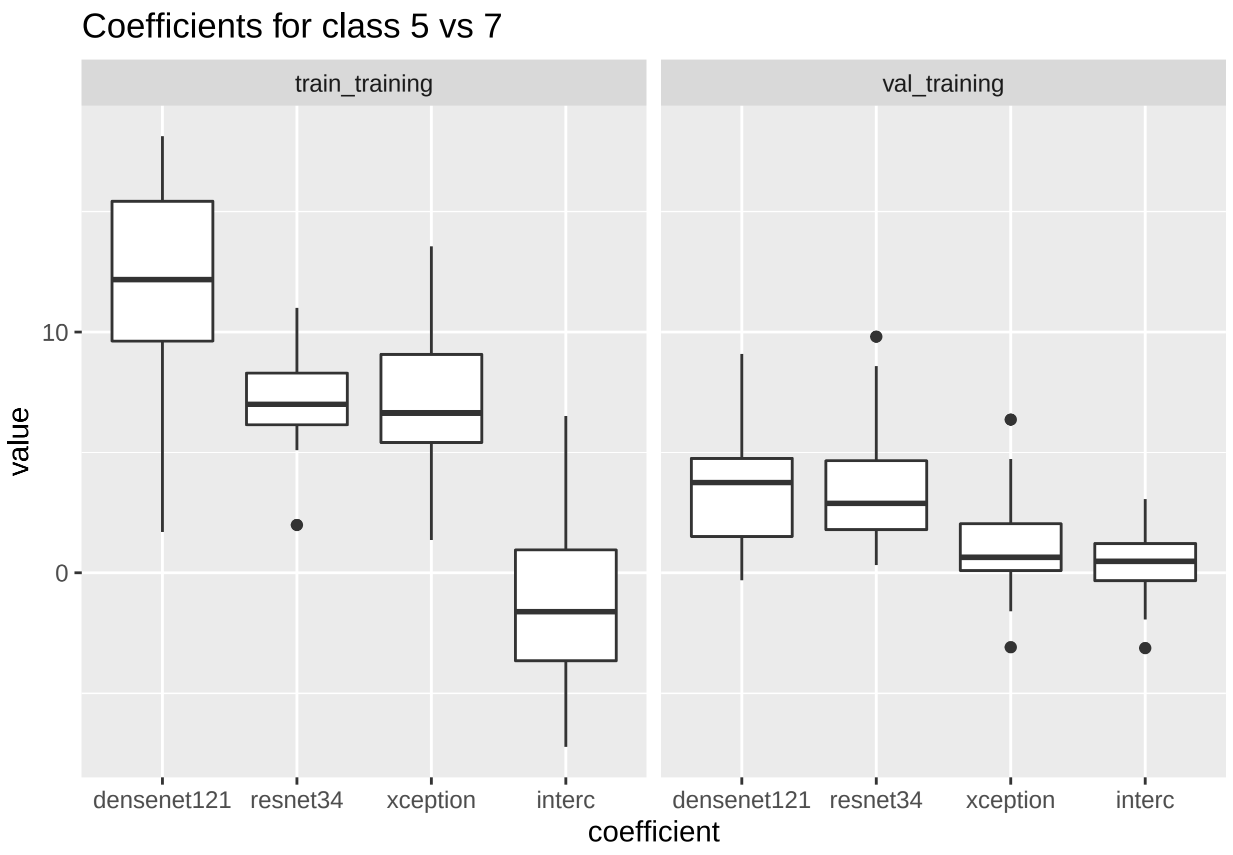

Coefficients for class 4 vs 7
train_training
val_training
value
20
10
0
−10
−20
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

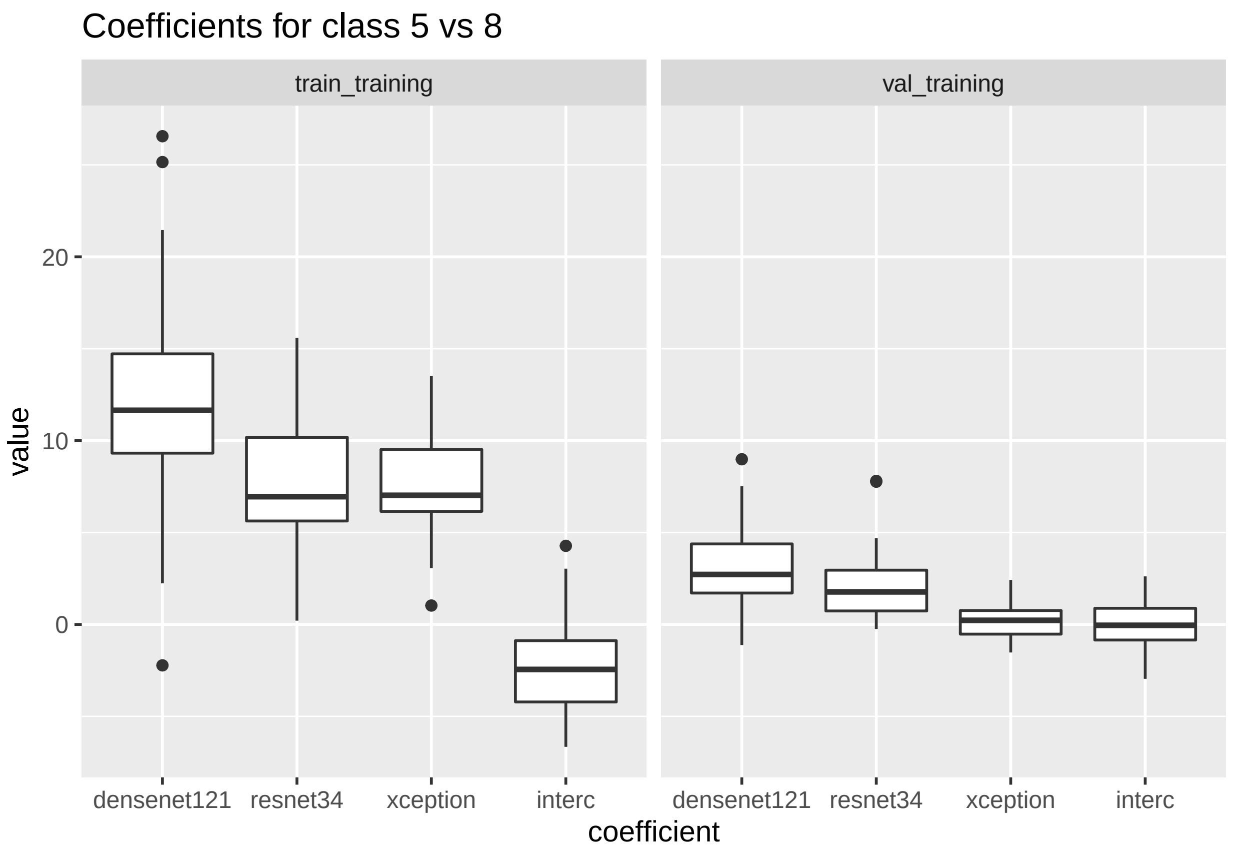

Coefficients for class 4 vs 8
train_training
val_training
value
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

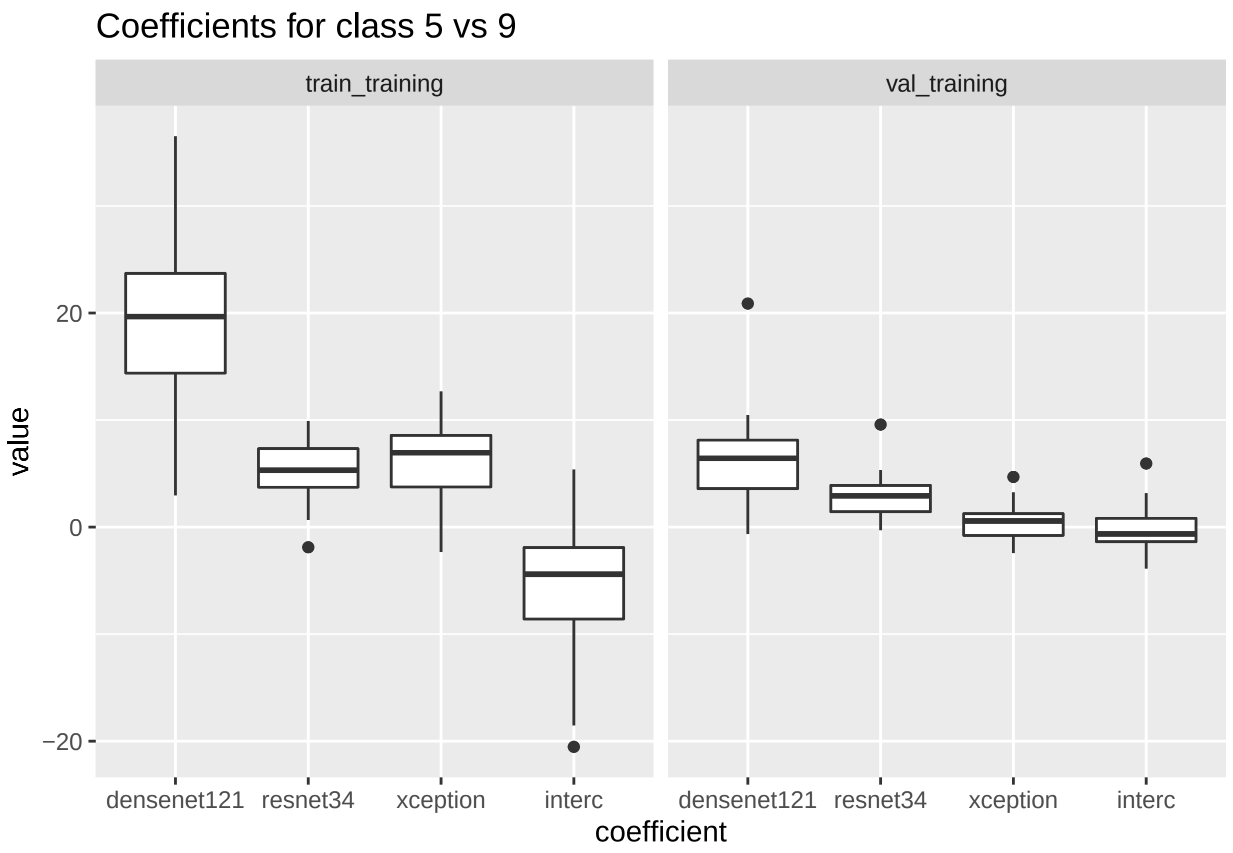

Coefficients for class 4 vs 9
train_training
val_training
value
coefficient
30
20
10
0
−10
−20
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc

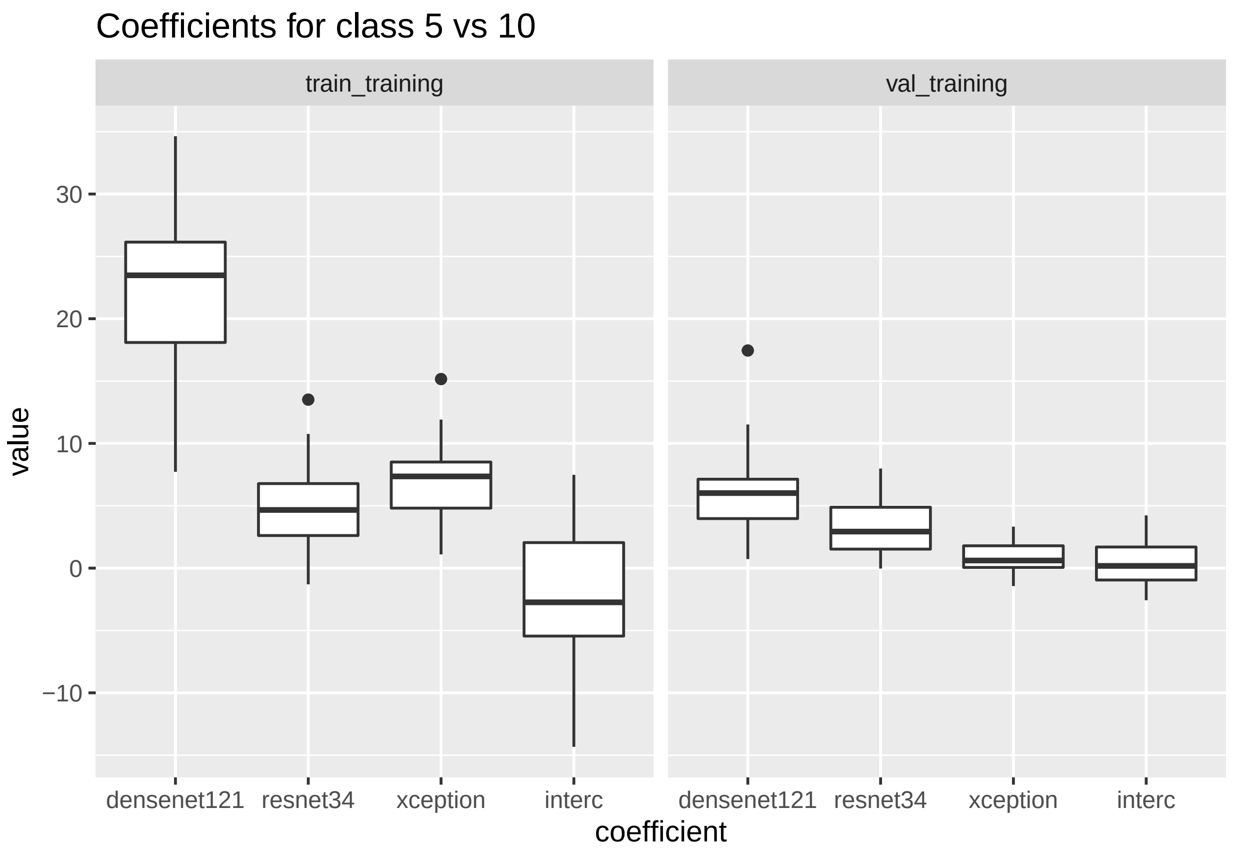

Coefficients for class 4 vs 10
train_training
val_training
20
0
−20
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

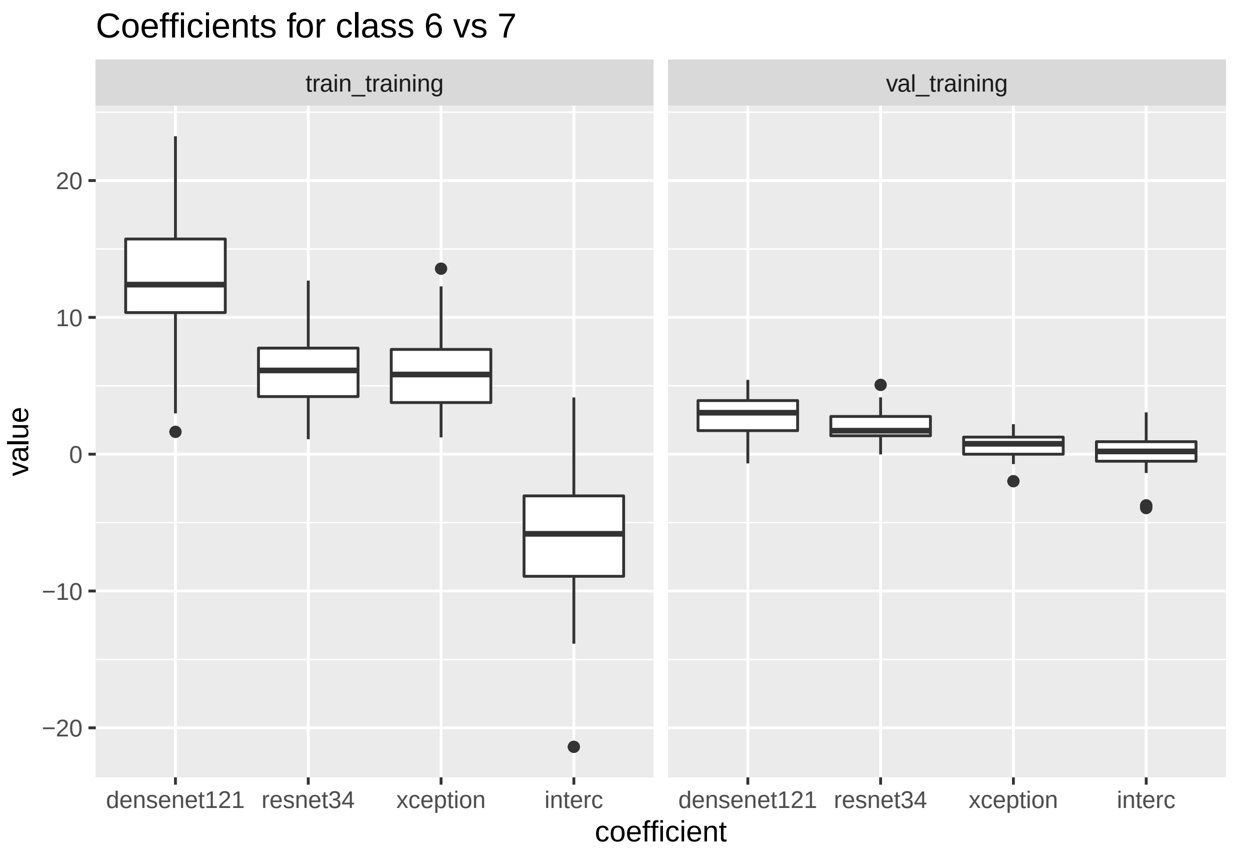

Coefficients for class 5 vs 6
train_training
val_training
value
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

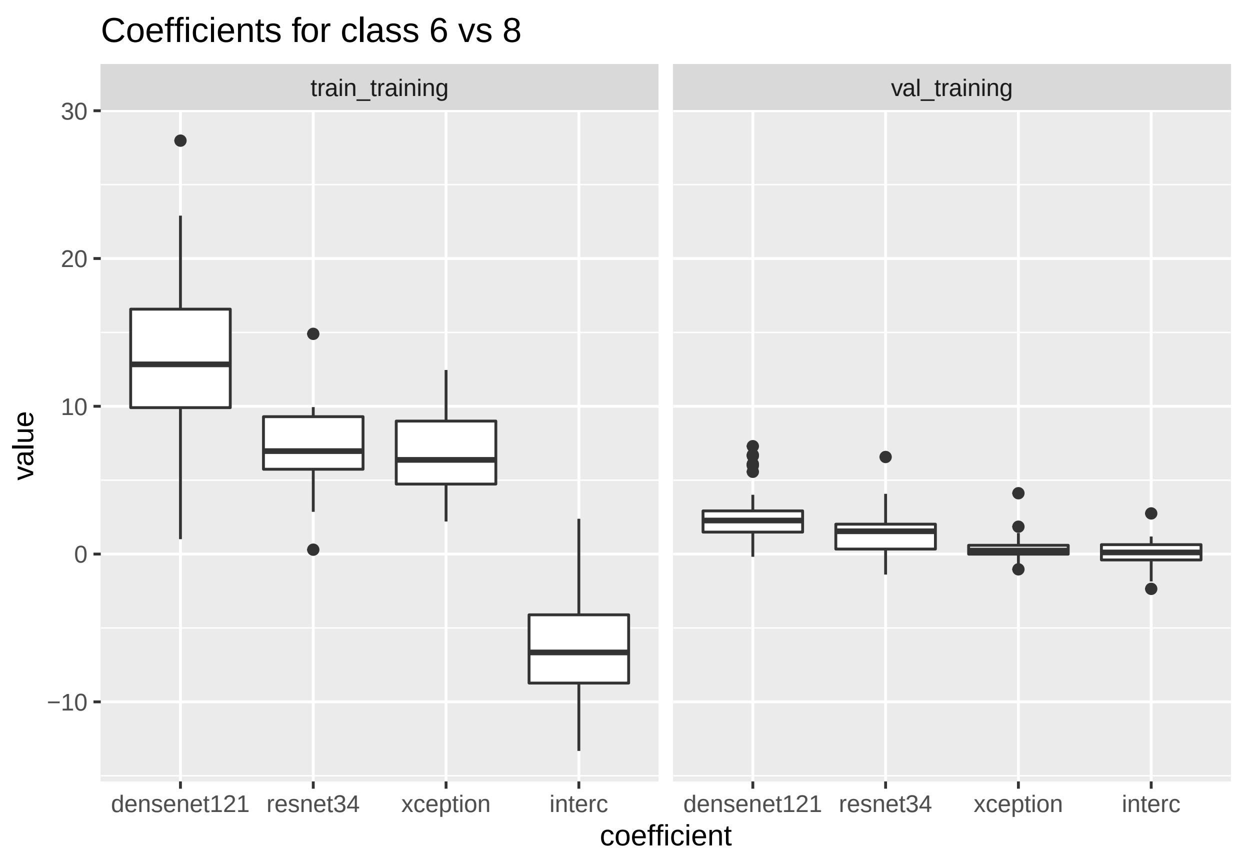

Coefficients for class 5 vs 7
train_training
val_training
value
10
0
densenet121
resnet34
xception
interc
coefficient

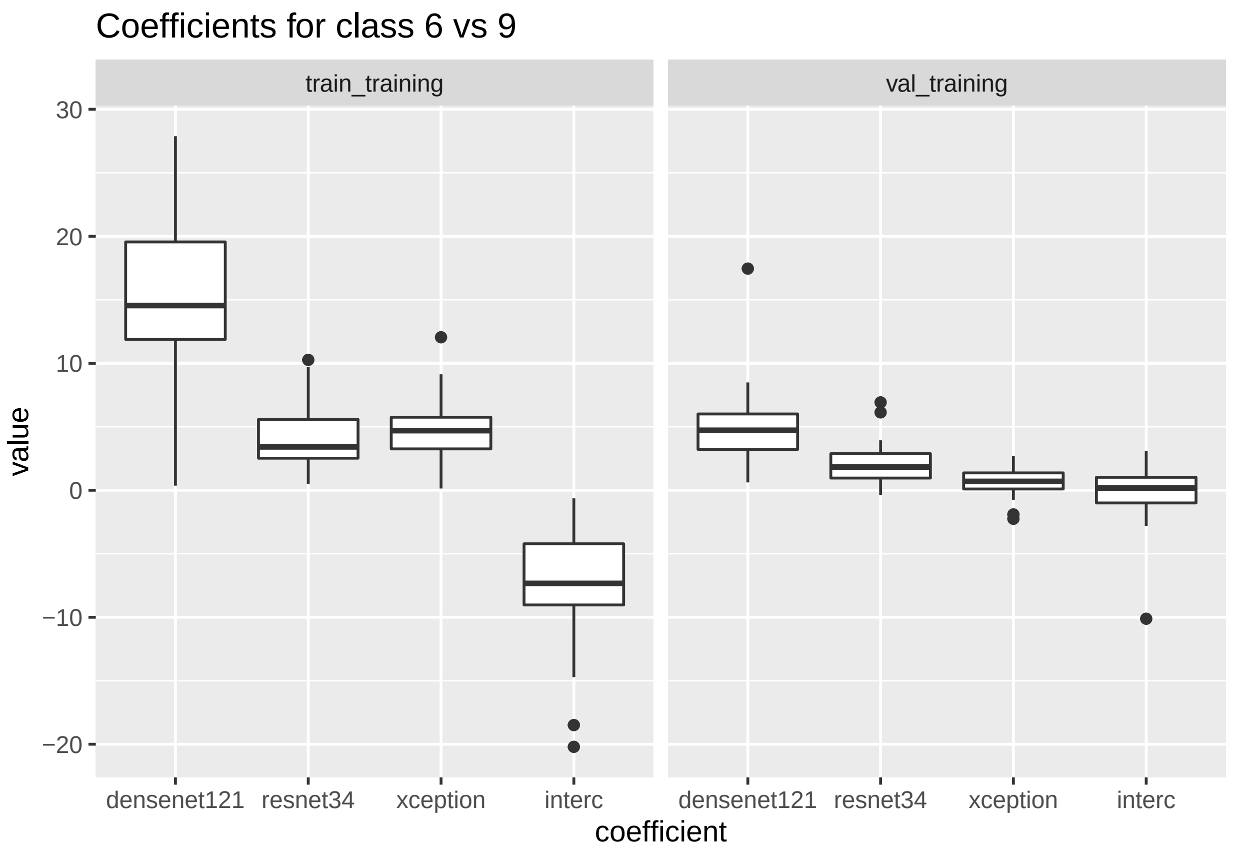

Coefficients for class 5 vs 8
train_training
val_training
value
20
10
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

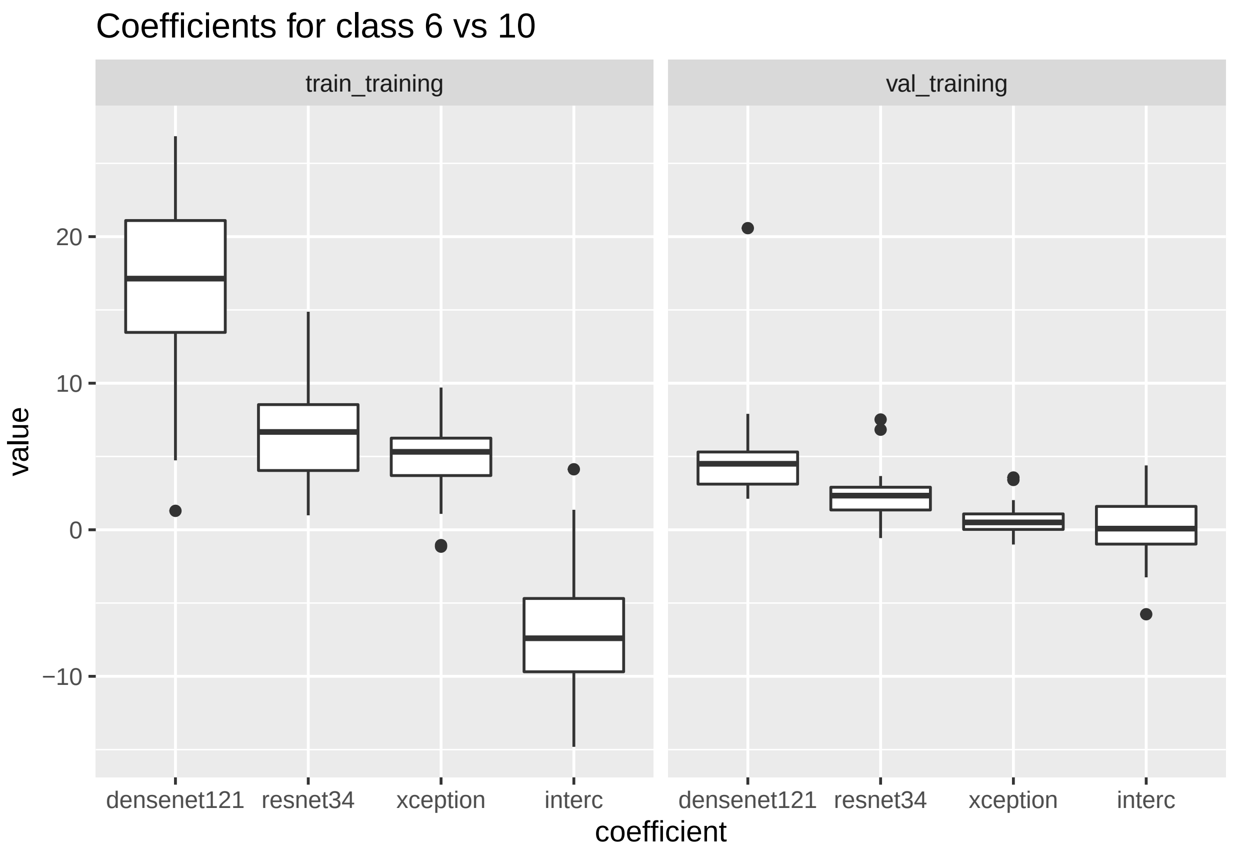

Coefficients for class 5 vs 9
train_training
val_training
value
20
0
−20
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

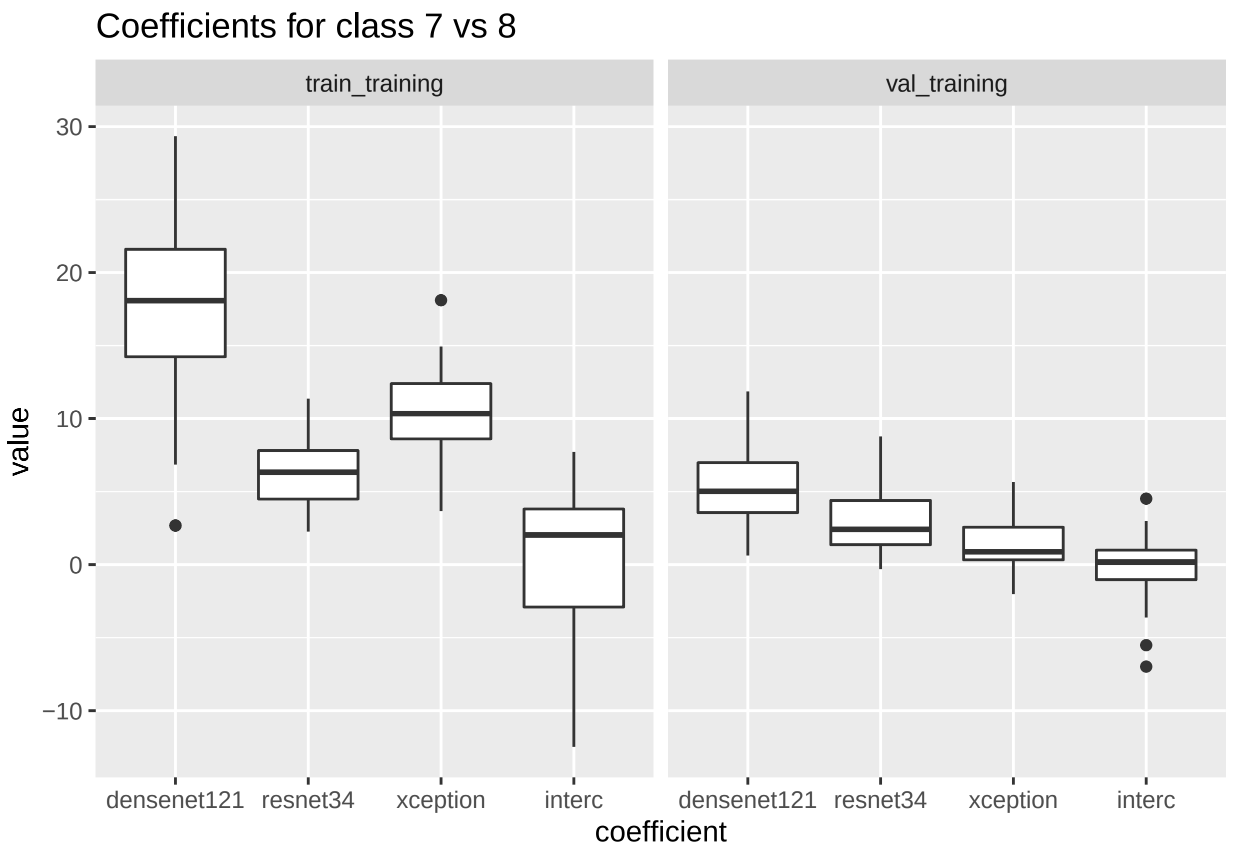

Coefficients for class 5 vs 10
train_training
val_training
value
30
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

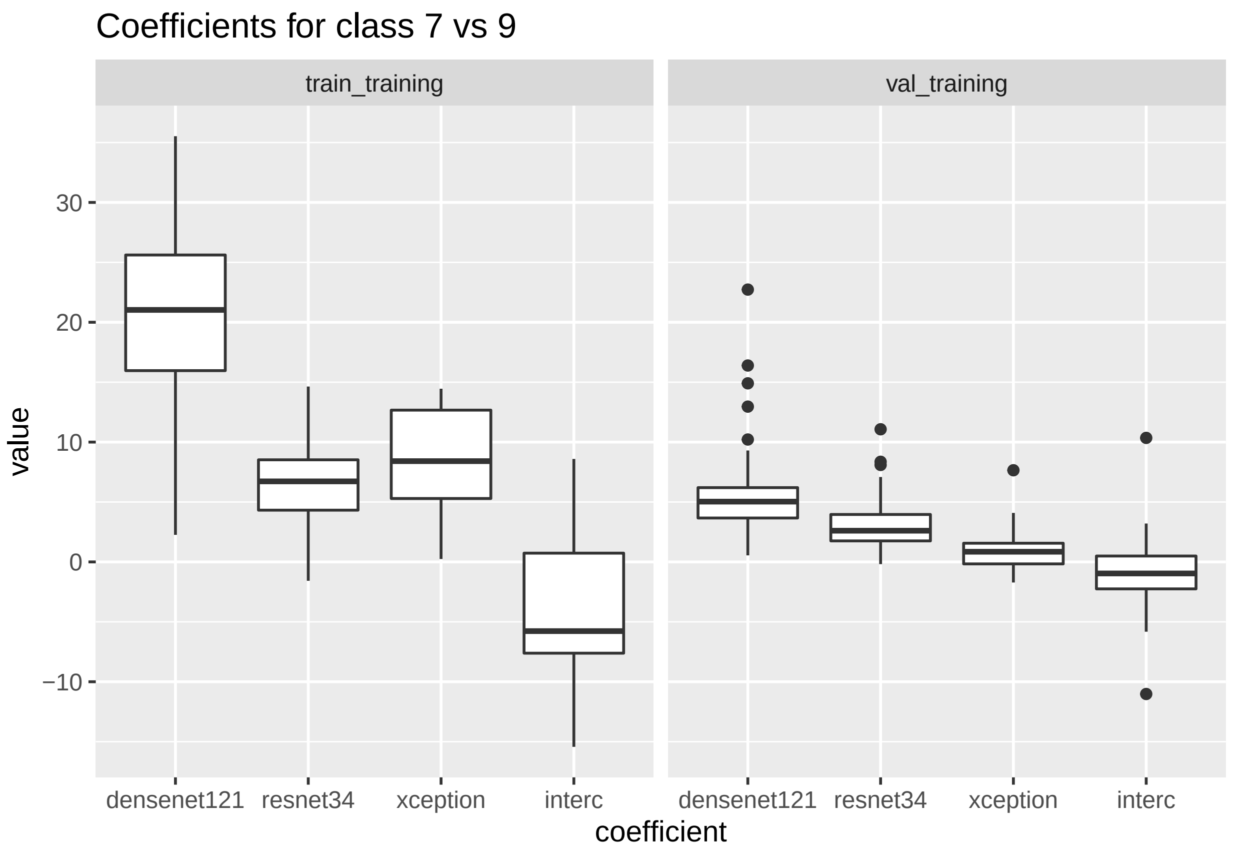

Coefficients for class 6 vs 7
train_training
val_training
value
coefficient
20
10
0
−10
−20
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc

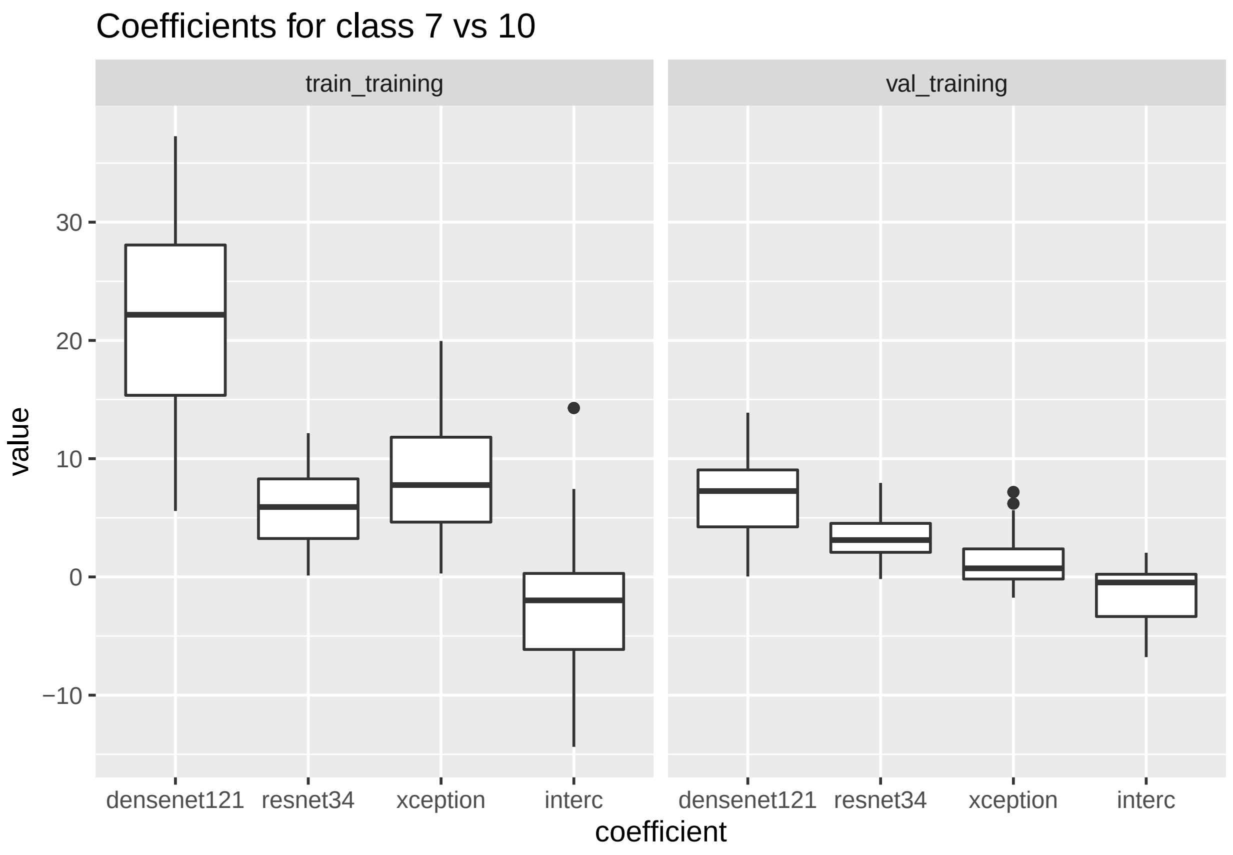

Coefficients for class 6 vs 8
train_training
val_training
value
30
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

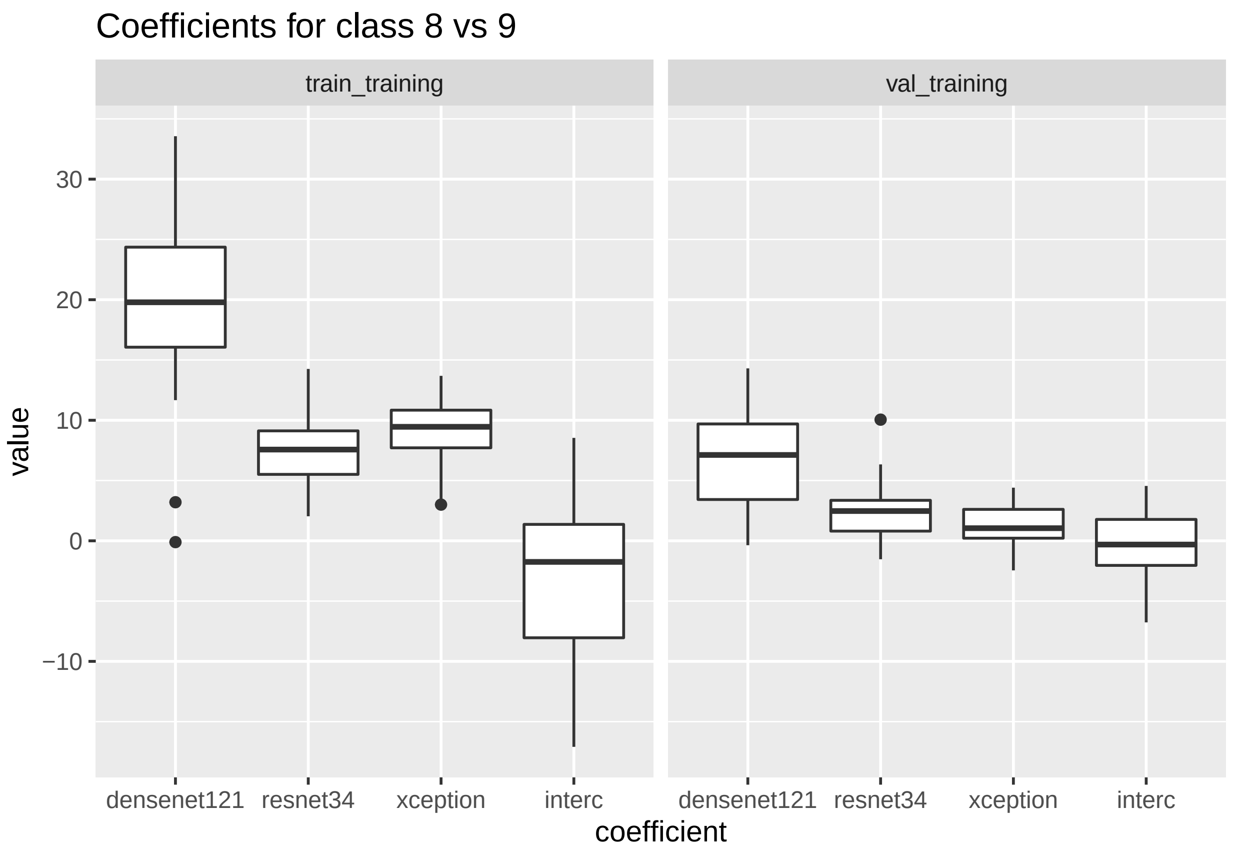

Coefficients for class 6 vs 9
train_training
val_training
value
30
20
10
0
−10
−20
densenet121
resnet34
xception
interc
coefficient

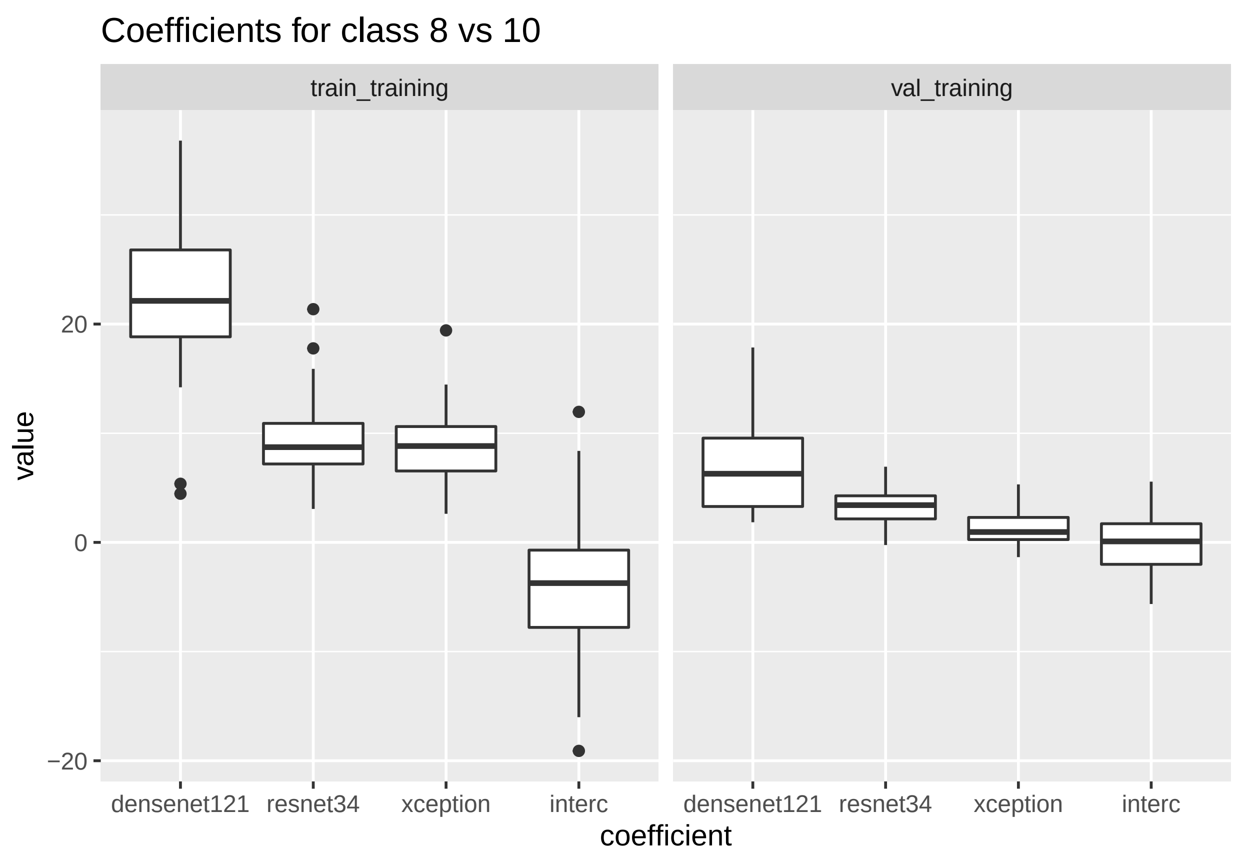

Coefficients for class 6 vs 10
train_training
val_training
value
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

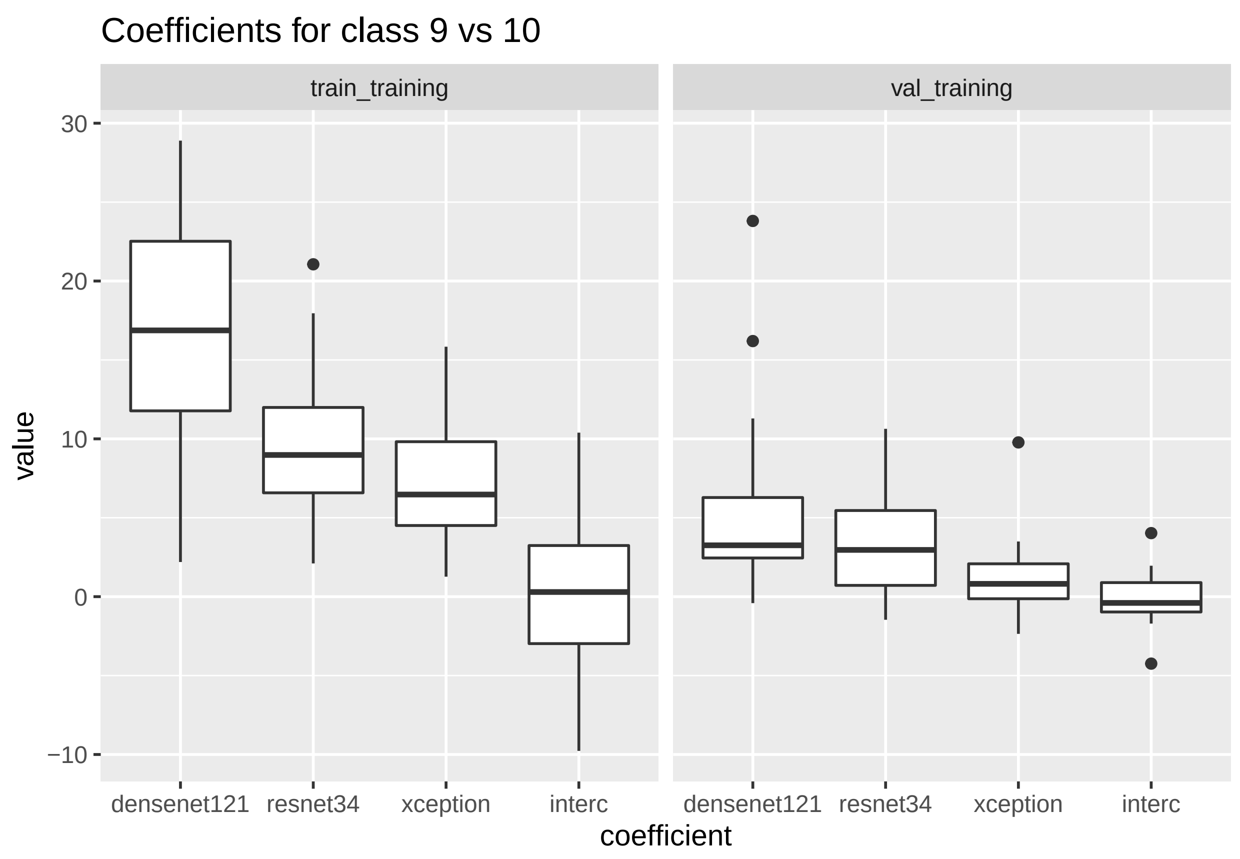

Coefficients for class 7 vs 8
train_training
val_training
30
20
10
0
−10
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 7 vs 9
train_training
val_training
value
30
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 7 vs 10
train_training
val_training
value
30
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 8 vs 9
train_training
val_training
value
30
20
10
0
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 8 vs 10
train_training
val_training
value
20
0
−20
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

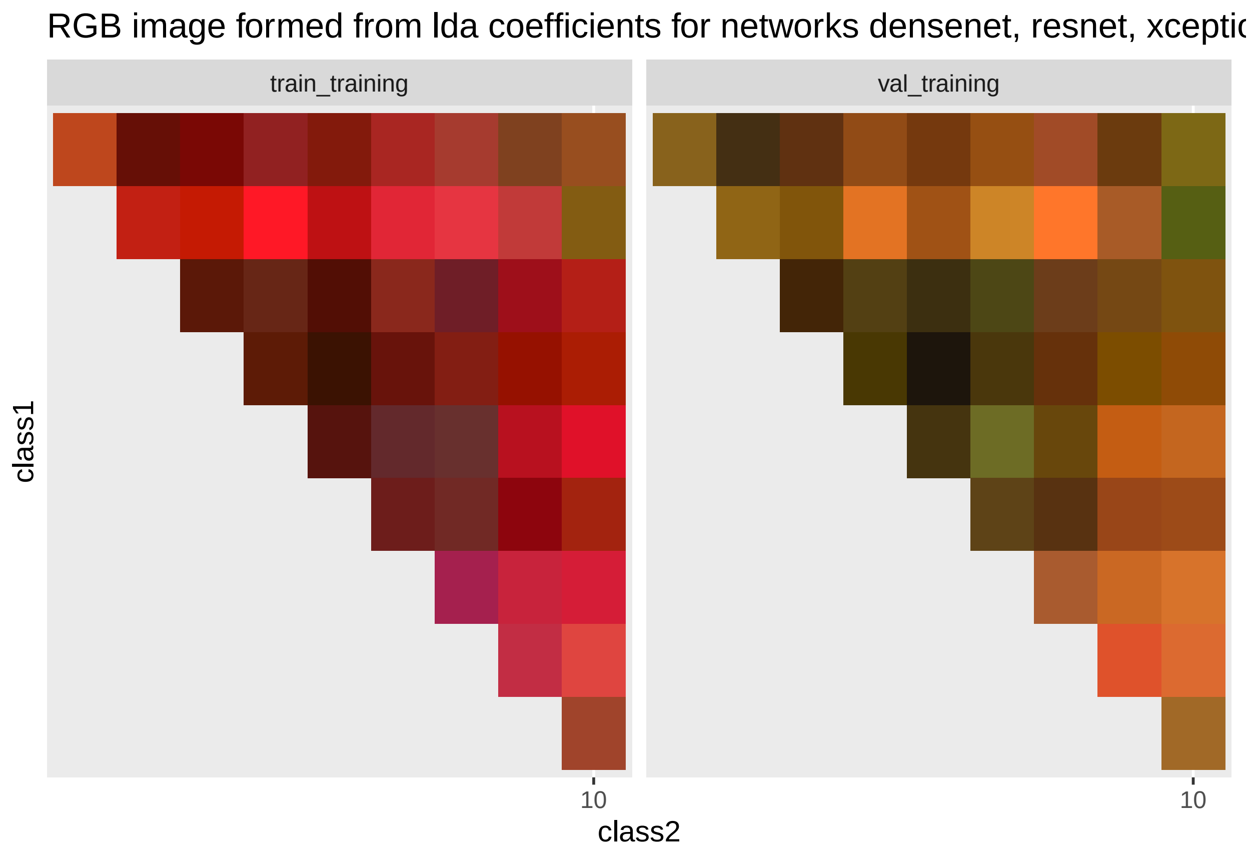


```
avg_lda_coefs <- lda_coefs %>% filter(coefficient != "interc") %>% group_by(class1, class2, precision, 
```

```
## `summarise()` has grouped output by 'class1', 'class2', 'precision', 'train_type'. You can override 
```

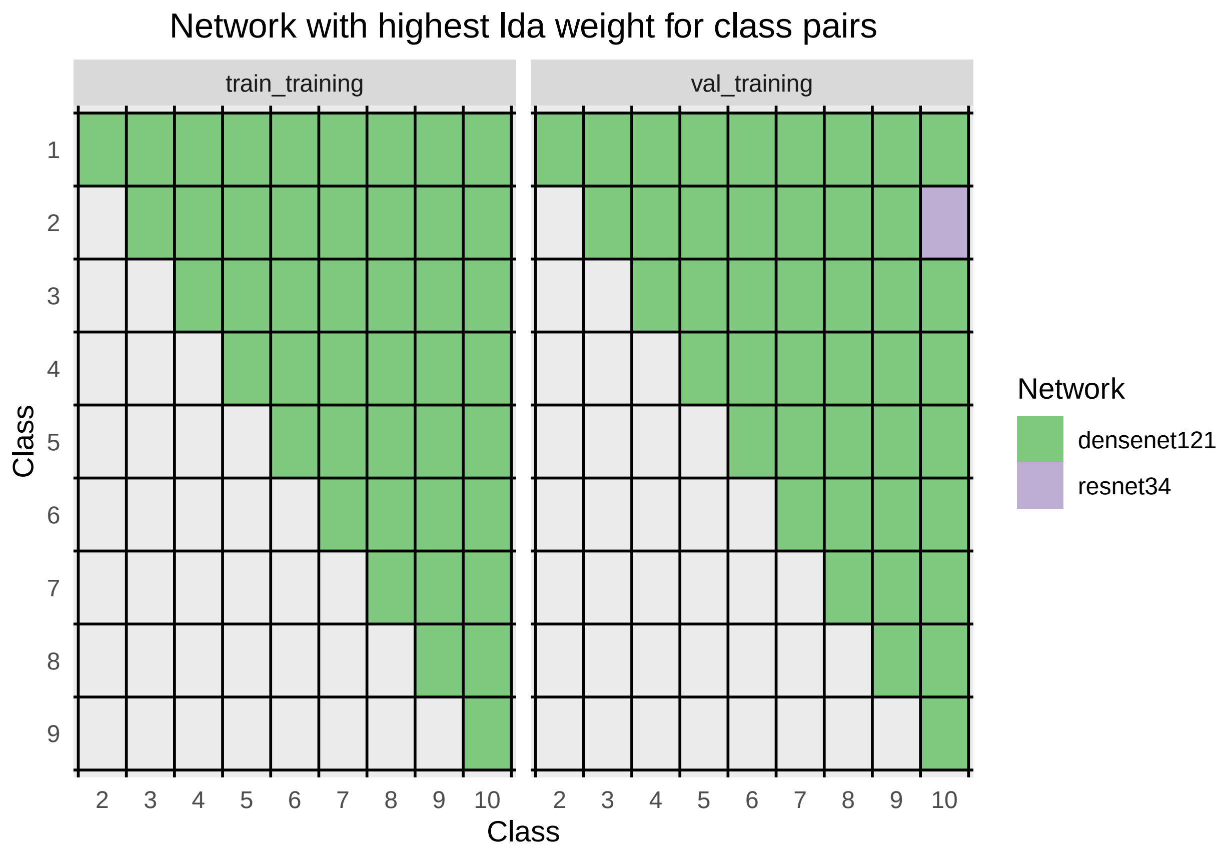
```
avg_lda_coefs_vt <- avg_lda_coefs %>% filter(train_type=="val_training")
avg_lda_coefs_tt <- avg_lda_coefs %>% filter(train_type=="train_training")
avg_lda_coefs_vt$value <- avg_lda_coefs_vt$value - min(avg_lda_coefs_vt$value)
avg_lda_coefs_vt$value <- avg_lda_coefs_vt$value / max(avg_lda_coefs_vt$value)
avg_lda_coefs_tt$value <- avg_lda_coefs_tt$value - min(avg_lda_coefs_tt$value)
avg_lda_coefs_tt$value <- avg_lda_coefs_tt$value / max(avg_lda_coefs_tt$value)
avg_lda_coefs <- rbind(avg_lda_coefs_vt, avg_lda_coefs_tt)
avg_lda_c_w <- pivot_wider(avg_lda_coefs, names_from = coefficient, values_from = value)
avg_lda_c_w[, c("class1", "class2")] <- lapply(avg_lda_c_w[, c("class1", "class2")], as.factor)
avg_lda_c_w$top_net <- factor(c("densenet121", "resnet34", "xception")[max.col(as.matrix(avg_lda_c_w[, 
```

```
raster_plot <- ggplot(avg_lda_c_w) +
  geom_tile(aes(x=class2, y=class1, fill=rgb(densenet121, resnet34, xception))) +
  scale_y_discrete(limits=rev, breaks=seq(0,classes, 10)) + scale_x_discrete(breaks=seq(0,classes, 10))
raster_plot
```



Correspondence between colors and networks is red - densenet, green - resnet, blue - xception.

```
coefs_grid <- ggplot(avg_lda_c_w, aes(x=class2, y=class1, fill=top_net)) +
  geom_raster() +
  scale_fill_brewer(type="qual") +
  facet_wrap(~train_type) +
  scale_y_discrete(limits=rev) +
  geom_vline(xintercept=seq(-0.5, 9.5, 1.0)) +
  geom_hline(yintercept=seq(-0.5, 9.5, 1.0)) +
  guides(fill=guide_legend(title="Network")) +
  xlab("Class") +
  ylab("Class") +
  ggtitle("Network with highest lda weight for class pairs") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())

coefs_grid
```



Densenet is clearly dominant for both LDA training methodologies.